Escuela
Politécnica
Superior

Development of a complete Optical Music Recognition workflow

Máster Universitario en Ciencia de Datos

Trabajo Fin de Máster

Autor:
Antonio Ríos Vila
Tutor/es:
Jorge Calvo Zaragoza
José M. Iñesta Quereda

Julio 2021

Universitat d'Alacant
Universidad de Alicante

Development of a complete Optical Music Recognition workflow

A complete Optical Music Recognition pipeline via Agnostic Transcription and Machine Translation

Autor
Antonio Ríos Vila

Tutor/es
Jorge Calvo Zaragoza
Departamento de Lenguajes y Sistemas Informáticos
José M. Iñesta Quereda
Departamento de Lenguajes y Sistemas Informáticos

Máster Universitario en Ciencia de Datos

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Motivation, justification and general purpose

My fondness with computers comes since I was young. My first steps into IT were with my father's old computer, which had a Windows 98 installed. Besides spending hours playing videogames, I enjoyed experimenting with machine's tools. The projects I made during these years were a website in Microsoft Pullitzer and a *mod* for a videogame, *Amnesia: The Dark Descent*, without previous programming experience. I enjoyed experimenting with computers. Admittedly, this hobby and my passion for videogames brought me to study Multimedia Engineering.

My interest in Artificial Intelligence and Machine Learning also appeared during college. In Devices and Infrastructures for Multimedia Systems, the teacher showed us a video where a machine learnt how to play Starcraft 2, a game which, even devoting hundreds of hours, I could never master. This curiosity stayed in me. Despite that, I never tried to advance on it. The reason was all of the concepts that I had to learn autonomously. I always saw this field of knowledge as something unreachable.

Last year, I worked on my final degree thesis: *ReadSco*, where I produced a user-oriented Optical Music Recognition tool to bring research results to the public. One of the steps that most interested me during the development of this application was the encoding step, which was indeed something yet unexplored by the Optical Music Recognition (OMR) research community. Even of producing a simple system to produce tangible results for the applications purposes, the Machine Translation research field caugh my attention, as I saw that it could be interesting to apply its techniques to the musical context, which could produce potential benefits to the research field. However, time constraints and work deadlines unabled me to dedicate time on diving into this application.

That is, during this year, I profited the opportunity to develop a final master's thesis based on this research line, where I could be able to work with technologies I was interested with (the Seq2Seq neuronal mechanisms and the Transformer architecture) and explore a pendent task that kept me interested during my last work.

Abstract

Optical Music Recognition, conocido como OMR, es un área de conocimiento que investiga la lectura computacional de los documentos musicales a partir de su imagen. A pesar de todos sus esfuerzos en conseguir procesos para avanzar en el reconocimiento de las partituras musicales, este campo de investigación tiene varios problemas para la puesta en práctica de los resultados que obtiene. Una de las razones es el poco tratamiento que le ha dado la comunidad científica, por motivos de gestión de prioridades, a un paso fundamental del proceso: la codificación, referida en la literatura como *encoding*.

Este paso en concreto consiste en convertir la salida de un proceso de reconocimiento gráfico, la cual está codificada en un lenguaje agnóstico (donde únicamente se representan facetas gráficas como la posición y la forma que tiene la nota en el pentagrama) a una codificación semántica estándar, la cual representa la partitura tal y como se entendería la música en sí (mediante un lenguaje que representa los roles sintácticos y semánticos de cada nota en el pentagrama). Es necesario obtener esta representación para garantizar la exportabilidad de los resultados producidos para procesarlos en otros programas de procesado digital de la música.

En este Trabajo Final de Máster, trabajaremos este aspecto para, finalmente, producir un sistema completo de reconocimiento de música basado en dos pasos: una transcripción agnóstica de la imagen de un pentagrama de entrada y un paso de traducción automática entre la salida de dicho reconocimiento (la cual está en dicha codificación agnóstica) a una codificación semántica.

En primer lugar, investigaremos más acerca de cómo enfocar el problema de la traducción automática entre lenguajes musicales, ya que es algo en lo que aún no se ha profundizado en literatura existente. Propondremos diversos sistemas, desde los estadísticos a los basados en redes neuronales profundas, para afrontar el problema y evaluaremos su rendimiento en distintos conjuntos de datos. Una vez realizado este paso, evaluaremos la factibilidad de estos sistemas y propondremos las distintas situaciones en las que un traductor automático entre dos codificaciones musicales pueden ser convenientes.

Después de esta investigación, llevaremos a la práctica los resultados obtenidos y crearemos el sistema completo de reconocimiento musical. En esta parte del trabajo, comentaremos cómo se realiza el reconocimiento gráfico de un pentagrama musical y explicaremos tanto los sistemas propuestos que usan traducción automática como el caso base, donde crearemos un sistema que no necesita traducción automática para producir una codificación semántica de la partitura reconocida (aunque tiene peores resultados que si se hace este reconocimiento mediante codificación agnóstica).

Finalmente, concluiremos comentando que, tal y como planteamos como hipótesis en este trabajo, existen ventajas en el uso de sistemas completos de OMR basados en traducción

automática, al igual que indicaremos las situaciones en las que este tipo de sistemas pueden ser útiles para la comunidad dedicada al procesamiento, la creación y la digitalización de corpus musicales.

Acknowldedgments

First of all, I would like to thank my supervisors, Jorge Calvo Zaragoza and José Manuel Iñesta, for all the support they have offered both since I started my journey on research, for teaching me everything I need to do this kind of work and, furthermore, for placing all their trust in the project. I would also like to thank David Rizo, Miquel Esplà and Pedro J. Ponce for their collaboration and support in the presented research. On the other hand, I would also like to mention my friends and colleagues. Starting with the latter, I would like to thank my research colleagues: Paco, José Javier and María, with whom I have shared all the journey done in this project and received their advice and support during all the process. Other friends that I cannot forget are those who, since the age of fifteen, have followed all my steps up to this point. These people are Javier, Iván and Adrián, who even though on many occasions they understood little or nothing about what I was talking about, they listened to everything I had to say and always showed me that life is something more than pure routine, that I do not have to bit myself about my problems. As they say, there are friends that become family, and they surely did.

Last but not least, I am obliged to feel grateful to my parents, whose patience, trust and perseverance have made it possible for me to be here and not to have thrown in the towel on any hard occasion. Sometimes I think I am not aware of all they have done for me.

Music expresses that which cannot be put into words
and that which cannot remain silent.

Victor Hugo.

Contents

List of Figures

List of Tables

1. Introduction

Music represents a valuable component of our cultural heritage. Musical composers have transmitted their work through time by two means: the word of mouth one, with popular tradition, and the written one, where the composition is transcribed in documents known as music scores. Thanks to this, we are nowadays able to read, explore and know music as it is. Music scores represent a fundamental piece of this possibility, as they allow the continuous reinterpretation of music works and warrant the stay of the original compositions.

However, these music scores were transmitted, more frequently in the historic music context, through handwritten documents, which get damaged over time. The responsibility for preserving these texts formerly fell to the clergy, who copied every page of these works by hand. This process, however, is nowadays unfeasible, as the amount of music works cannot be processed manually in a reasonable amount of time. The problem has been alleviated thanks to computer science, whose capabilities of automation and accuracy bring useful results to ease this tedious work. Indeed, it is common to see that diverse areas related to the cultural heritage preservation work with automatic tools to achieve their goals. For example, in plain text libraries, they use Optical Character Recognition (OCR) and Handwritten Text Recognition (HTR) techniques and systems to scan and recognize these texts. Analogous to these technological areas, it is reasonable to deduce that music transcription can also benefit from automatic processes. From this premise, we enter into the field of Optical Music Recognition.

1.1. Optical Music Recognition

Optical Music Recognition (OMR) is a research field that investigates how to computationally read music notation in documents (Calvo-Zaragoza et al., 2020). Specifically, it is a research field which joins the capabilities of knowledge areas like Computer Vision, Machine Learning or Digital Musicology to perform the recognition and the digitizing of music scores.

The first question that may rise when one is introduced to this research field is in which terms it is different from the text recognition area (which involves OCR and HTR), whose goals are similar. In fact, OMR has been frequently referred to as "OCR for music" (Burgoyne et al., 2008). This definition is inaccurate due to several details that music hinders itself.

First music has a dual nature. A musical note is interpreted and identified by basically two features: its shape and its position in the staff. The first one indicates the duration of the sound and the second one defines its tone, with other conditioners such as, for example, the key and the nearby alterations. This nature not only exposes how a musical note has to be interpreted as it also shows how it relates to the rest of the components in a staff. In fact, the analysis of these properties is a particular computational challenge that has been addressed as a specific challenge (Byrd & Simonsen, 2015), so it is reasonable to discriminate OMR from other research fields since it requires the approach of specific challenges that rise from its nature.

Another reason for considering OMR aside from other knowledge areas is in the application domain itself. In OCR and HTR, for example, the main objective is to obtain the ordered recognized words from a document image. However, music has to go beyond this goal. Representing music notes just as positions is clearly insufficient to retrieve their meaning. However, this interpretation is also highly dependent on conventions, which can change depending on the historical and social context where the music document was produced. This is something that happens frequently when addressing different music genres, such as jazz or classical music and epochs, where the notation type and style differs. For instance, an articulation is not interpreted the same way in the baroque or romantic period, or a half note is interpreted as a *minima* in mensural notation. As it can be observed, OMR requires providing results in an abstraction level that text recognition does not have to. This leads this domain to be more ambitious than text recognition, since there is an interpretation layer to the recognition process that does not have good analogy in other natural languages. This fact leads the community to bring new solutions, both technical and linguistic, to produce those complex results that OMR requires digitizing music scores.

Once the particularization of the OMR is justified, the question remains as to which processes or sub-areas make up this knowledge field. The techniques used in this research branch have been polished and adapted thanks to technological progress, specifically Machine Learning. However, its base is funded in four main blocks (Calvo-Zaragoza et al., 2020; Bainbridge & Bell, 2001; Rebelo et al., 2012): pre-processing, music object detection, notation assembly and encoding. A complete OMR process has to cover these four fundamental pillars, which is the main premise that we review during this work. The input for these systems consists of a musical score, either handwritten or printed. The output is the digital version of it in a standard format, such as MEI (Hankinson et al., 2011), MusicXML (Good & Actor, 2003) or Humdrum **kern (Huron, 1997).

Once given this brief description of OMR, we discuss the recent advances in OMR and the topic of interest that this work approaches.

1.2. Recent advances in OMR and the need of the Encoding step development

In this section, we overview the recent advances OMR has and the current open issues and challenges that have not been approached yet.

Most of the existing OMR literature is framed within a multi-stage workflow, with steps involving image pre-processing—including staff-line detection and removal (Dalitz et al., 2008)—symbol classification (Pacha & Eidenberger, 2017), and notation assembly (Pacha et al., 2019). Recent advances in Machine Learning, namely Deep Learning, have changed the way OMR is approached. Instead of using legacy pipelines, which were based on statistical and data-driven approaches, the research community has taken advantage of the capabilities of neural methods, which converge some of these stages into single ones and present better and more straightforward implementations. On the one hand, we have the segmentation-based approach, where the complex multi-stage symbol isolation workflows have been replaced for region-based CNN that directly recognize symbols in a music staff (Pacha et al., 2018; Tuggener et al., 2018).

On the other hand, there is the end-to-end recognition, which retrieves the symbols within

a staff without any previous segmentation or isolation. Specifically, we find solutions based on Convolutional Recurrent Neural Networks (CRNN) that come in varying configurations: some with the so-called Sequence to Sequence (*Seq2Seq*) (Sutskever et al., 2014) architecture (van der Wel & Ullrich, 2017; Baró et al., 2020), and also those trained using the Connectionist Temporal Classification (CTC) loss function (Calvo-Zaragoza et al., 2019; Sánchez et al., 2019), which currently are the state-of-the art systems in this area.

Despite the recent advances that we have presented, many issues hinder taking OMR results from research to practice, such as the file management system, the development of appropriate visual interfaces, or exporting the recognition results to standard formats, which is currently our subject of interest in this work.

One of the main challenges in OMR analysis is to define representation language to define the recognized symbols in the networks output. A musical note contains different meanings depending on some determinants and variables, so it is challenging to find a suitable encoding that could fit those semantic needs without having an extensive dictionary. A possible solution could be to produce a custom encoding that describes easily these relations and does not have an extensive vocabulary. However, it is not as practical as it may seem. In (Calvo-Zaragoza & Rizo, 2018), it is discussed the possibility of using another music representation based on a different look to music. There exists a music notation which avoids considering the analyzed note in musical terms. Whereas, it sticks to the physical features. Instead of determining the pitch of the note, it presents the shape of the recognized symbols and their position in the staff. If this token sequence is encoded from left to right, we obtain a valid music score representation without taking into consideration the complexity of the music semantics. In other words, the recognition is done in an agnostic way by just describing the graphical features of the analyzed piece. This specific notation is referred to as the *agnostic* representation.

Typically, the existing DL-based approaches described before cover all the processes that involve the transcription of an input image, which is usually a presegmented music staff, into a text sequence use this graphical representation.

However, even obtaining such descriptive sequences with that simplicity, these results cannot be used by an end-user or reproduced in a music tool or visualizer, as there still is required to recover music semantics as well. This last step to achieve an actual digital score is the so-called encoding process, where the graphical recognition outputs, without specific musical meaning, are converted into a standard semantic encoding. Unfortunately, this step is hardly addressed in the OMR literature, due to the focus the challenges of the previous steps require.

There exists little literature that approaches this step with rule-based or grammar systems (Rossant & Bloch, 2006; Couasnon, 2001). However, these solutions present issues in terms of scalability, as new rules have to be manually introduced if the source or the target encodings are updated, which make the maintainability of the system challenging to handle as the music encodings grow. Recent work on the topic has shown that the application of machine learning techniques, specifically machine translation (MT) approaches, may be a viable solution to solve this problem without scale issues, as the encoding step has close resemblance to challenges that have been faced in machine translation scenarios. We explored this topic in previous work with a simple end-to-end model. In (Thomae et al., 2020) it was used a non-standard simplified semantic encoding to explore the feasibility of applying these techniques to the encoding step. However, even of proving good results, this work only scraps the surface

of the topic, as the output of the system is not encoded in a standard semantic format, so we cannot consider that the topic has been yet fully addressed, as there is currently no research focused on the feasibility of performing an encoding step between the agnostic encoding and a standard semantic one. In addition to this, even of being proved as a doable task, there is also no proof that completing the pipeline with this encoding step can be applied to a real-case scenario, where the graphic recognition errors have to be handled. We can say that there is still work to do in order to establish a solid knowledge about applying the concepts of the NLP area to perform the OMR encoding step.

In this work, we perform our research on this topic. We address both of the issues mentioned in the previous paragraph: the study on the feasibility of performing Machine Translation between the agnostic and a standard semantic encoding and then, with the produced results, we research on applying the learned techniques to the encoding step in an OMR process. In other words, we put our research into practice, and we analyze if the produced results by this novel formulated pipeline produce any benefits in comparison to a direct encoding (which does not take into account) approach, which is lately described in the following chapters. Once done this study, we would establish an initial point for further research with a clear idea on how could be these complete pipelines be approached.

1.3. Research goals

One fundamental step in this work before diving into research and implementation is to define the objectives to be achieved during the development of this complete OMR pipeline.

Our main goal in this project is, as we have mentioned before, to produce a complete OMR pipeline which uses a two-step approach:

1. Performing a graphic recognition which takes an image as an input and outputs an agnostic encoded sequence of the recognized symbols. This system is already implemented in the state-of-the-art OMR systems, so it will be introduced when it is used during the development process.

2. Performing a Machine Translation process between the outputted graphics-based language and a standard semantic encoded sequence.

To achieve this goal, we divide it into two main objectives, as we have already explained before. First, it is required to research and evaluate the application of Machine Translation techniques when performing the encoding step.

This first main goal can be depicted in the following sub-objectives:

- To study the most suitable semantic encoding to perform the translation process, as the musical area offers several semantic representations of a music score.

- To identify the potential challenges and conflicting musical concepts that have to be tackled when performing a translation between musical languages and group them into categories to identify them.

- To summarize the possible ambiguous or corner-case situations that can be present in a musical corpus and can be hard to solve even for humans without any musical rules knowledge.

- To propose several Machine Translation models and configurations to perform the task (which have been theoretically presented in the state of the art chapter).
- To evaluate the performance of the models in several corpora with different features to establish the advantages and the limitations of the proposed approaches when performing the task.
- To study, based on the results, the feasibility of performing an Automatic Machine Translation process between musical languages and, if so, discuss the best approaches that can be taken depending on the corpus features.

Once tackled this goal, the next part of the work focuses on bringing the produced systems from the previous work into a real-case scenario, where a complete OMR pipeline is designed with the two-step perspective (an agnostic transcription first and the machine translation after). This goal can be depicted in the following main points:

- To implement the complete OMR system with the two-step approach. There will be as many pipelines as models produced during the previous work
- To establish a comparison baseline with a system that may not need these approach (a direct encoding one, for example).
- To evaluate the performance of the models in several corpora with different challenges level to determine the limitations of the systems in complicated scenarios (which will be the most real ones).
- To determine, based on the experimentation results, if the proposed complete OMR pipeline based on a two-step process is currently beneficial for the OMR processing or does not improve the already established results by the baseline pipeline, which does not require this additional step.

Once described and established the main goals to be accomplished during this project, we can start our research process and work on the target tasks.

The project is structured as it follows: Chapter 2 presents and describes the main systems and technologies used for this work, which are discussed from a theoretical perspective. Once presented the theoretical background we are working in, we specify and describe in Chapter 3 the used tools and systems in this work, as well as we detail the labeling and data curation process for our experimentation. In Chapter 4, the first sub-objective of the project is addressed, where we explore the application of Machine Translation techniques to musical languages to perform the encoding step. After obtaining the results of this experimentation, we then apply these into practice in Chapter 5, where we create a complete OMR pipeline based on the state-of-the-art agnostic transcription system and the presented Machine Translation models. We then draw our conclusions, based on the obtained results from this previous chapter, in Chapter 6. We also discuss how this research can be extended and propose new topics that can be addressed in future work.

2. State of the art

Before diving into the development of the solutions that we propose to tackle the confection of a complete OMR pipeline, we consider it is essential to give a brief introduction to all the technologies that will be used. In this chapter, we review the theoretical concepts required to understand the underlying technologies of our research.

2.1. Brief introduction to Deep Learning

Deep Learning, known as DL, is a specific branch of Machine Learning, which focuses on the development of algorithms inspired by the structure of the human brain. The main objective of DL is to train a computer to solve problems through learning by example. This field of research has gained popularity in the technology sector over time. The main reason for this can be summed up in one word: accuracy. Nowadays, DL systems are capable of performing tasks with greater accuracy than humans themselves. Given the circumstances, it is common to observe that current development and computing trends are focused towards this area, as it somehow constitutes our technological future. Nowadays, DL is critical to advance in fields such as automatic driving, medical research, aerospace industry, industrial automation and electronics.

If we search more in-depth into Deep Learning, we will find a generic function that is also popular nowadays: the neural network. Before specifying the structures of interest in this project, it is worth asking ourselves a question: What is a neural network, and how does it work?

2.1.1. What is a Neural Network and how it learns

An Artificial Neural Network (ANN) is a generic function with a graph structure capable of, given an input, learn to generate its corresponding output. This inner learning consists of the generalization of the results of all the examples seen in order to predict inputs never received before. To better understand these systems, it is useful to study the most straightforward unit of such functions: the neuron, known in this field as the perceptron.

The perceptron is a structure conceived by Frank Rosenblatt to represent the behavior of a binary classifier (Rosenblatt, 1958). Its fundamental structure is depicted in Figure 2.1. As can be seen in this diagram, the perceptron is composed of several layers: the inputs, the weights, the sum operation and, optionally, the activation function. Each entry is multiplied by its assigned weight. The result of all these multiplications is then added up to give a single output to the perceptron. The latter result can be controlled by a bias, which determines which quantities we consider the neuron should give to contribute to the overall count,[1] and by an activation function, which grants the result with a non-linear property that helps

[1] It works as a minimum threshold to determine if the result is important or not to take it into account

to obtain more complex activations required in most of the demands problems for these structures. A basic neural network is itself a combination of layers composed of perceptrons. These networks contain three types of layers: an input with as many neurons as data units are contained in a sample, an output which will have as many neurons as categories to classify or values to predict, and the hidden layer, which processes the data to obtain an output according to the input received. Knowing this structure is essential. However, there is still a question to be answered: how does a neuronal network learn? It all comes down to a calculus matter.

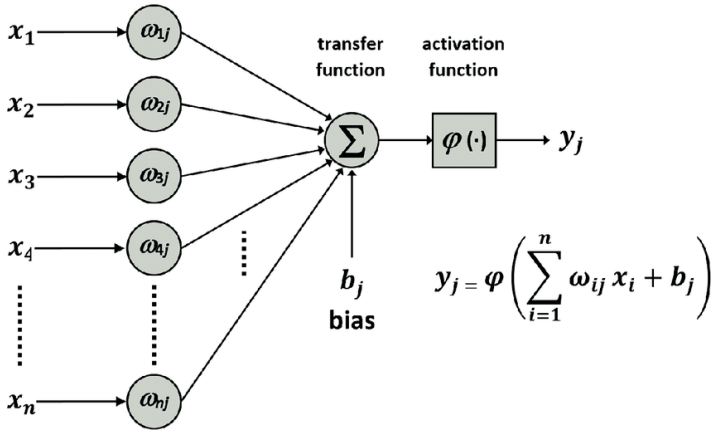


Figure 2.1: Basic structure of a binary perceptron

As we have seen in the structure of the perceptron, there are two variables that alter the result we obtain within its calculation: the weights assigned to each input and the bias applied to the sum operation. When a neural network is initialized, all of these variables are randomly set for each perceptron. It is foreseeable that, during the first training epoch, the network will fail notably in its predictions. Here comes the first essential question for understanding machine learning: How do we measure its deviation from the real outcome? Commonly, for each training sample, we get an error. There are several processes to obtain this result. The error increases when the network predicts results which are farther away from the expected ones and vice versa. Once all of these errors are obtained, we can process, for example, their average to know exactly how accurate the system is.

These described functions are known in the area as loss functions. Unfortunately, it is hard to represent them visually, as they typically handle millions of parameters. However, to understand the keys of this learning system, we will simplify the method with a function that uses only a single neuron loss. This representation is depicted in Figure 2.2. If we look carefully at the function described in the first picture, we realize that it has minimums. In mathematics, a minimum is a point in the function where the first derivative results in 0 and the second derivative is greater than 0. These values, in our context, are interesting, since they represent function points where the neural network will have minimal losses. Our goal is to try our best to reach these local minima. Unfortunately, finding exactly the minima of these functions is unattainable, as they handle millions of parameters. The strategy used for these cases is to find out in which direction we should descend to get closer to the desired point. On this occasion, it is easy to find the result, since it is provided by the slope of the first

derivative of the function, represented in the right side picture of Figure 2.2. As we descend, we will get closer to a local minimum of the function and, therefore, the neural network will be more accurate in its results. The process described by this simplification is known as gradient descent. Keeping this simple example, we observe that machine learning actually tries to solve non-linear optimization problems with multiple data and inputs. However, in real-case scenarios the situation is more complex than the basic example we have given, as we are dealing with more variables and parameters. It is required to use non-linear optimization methods (Aragón et al., 2019) in order to handle this situation. These methods basically take further the concept of the derivative and, as we are dealing with multiple parameters, extends the calculus to the partial derivatives of each variable that contributes to the model. These partial derivatives are referred in the bibliography as gradients, and they are key in determining the descent direction that we are interested to follow in order to obtain optimal solutions. How these gradients are handled depends mostly on the optimization algorithm taken into account, being Stochastic Gradient Descent or the Adam method (Ruder, 2016) the most popular among the theory of DL. However, the concept remains the same: we try to find a local minimum of the loss function of our problem to obtain less erratic predictions.

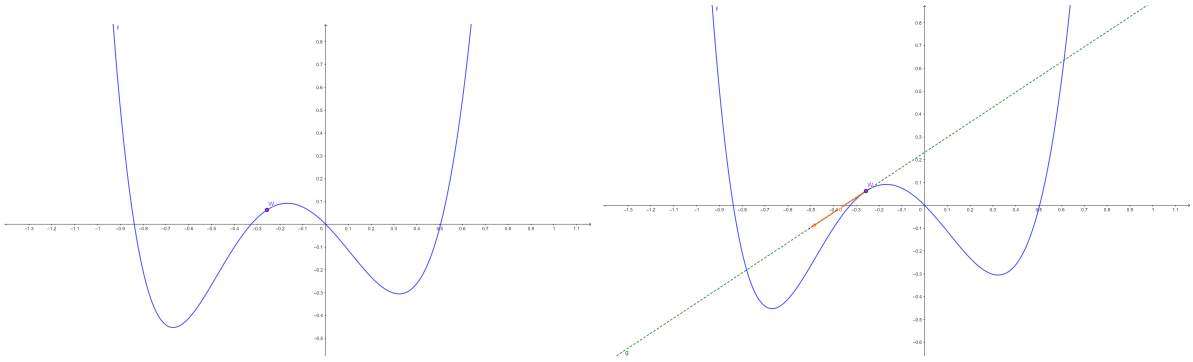


Figure 2.2: Simplified visual representation of the gradient descent process in only one dimension.

We know how to get the quality of the results provided by the network and what decisions have to be taken to improve them. The following question to this matter is: how do we apply the gradient descent to this complex function? As we mentioned before, the perceptron performance depends on two parameters: the weights and the biases. In essence, our main task will be to tweak these variables to obtain the desired results. The method for modifying them is known as the back propagation algorithm. This procedure, in a simple way, is in charge of, for each training sample, calculating how much the weights and biases of the perceptrons should be updated in order to obtain a more significant decrease in the cost function. Once the results are obtained, an average of the adjustment indicated by all the samples is calculated and applied to the current layer. This algorithm is executed recursively for all the layers of the network. Therefore, in the next step of the training, the results should be better. In mathematical terms, it is applying the inverse chain rule of the function computed in the specific layer. In the computational plane, this algorithm can be expensive, so in practice, an average of all the training samples is not performed. However, the set is divided into batches to find a close approximation to the optimal descent without wasting

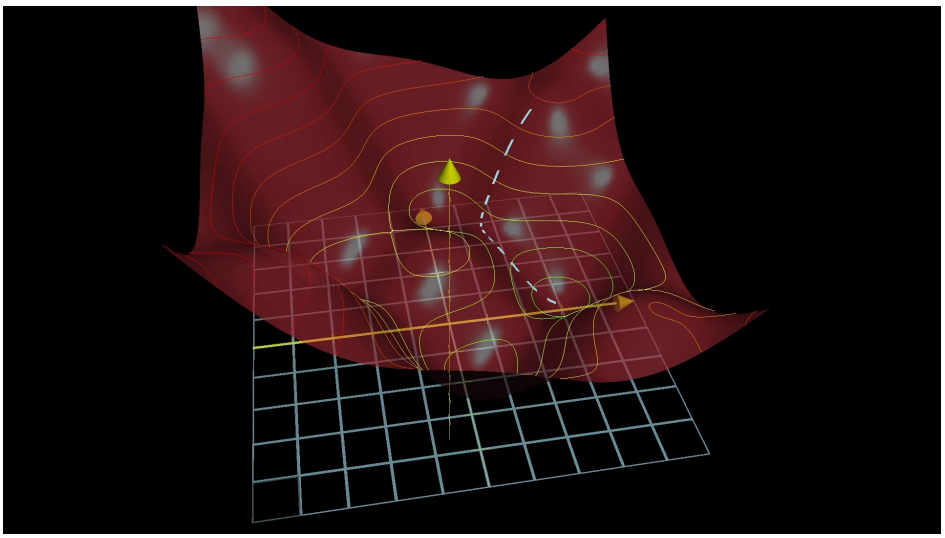


Figure 2.3: 3D representation of gradient descent taken from 3Blue1Brown's video series about Deep Learning

computational resources.

2.2. Convolutional Neural Networks

Convolutional Neural Networks, which we refer to as CNN, are Artificial Neural Networks specialized in detecting the features of input signals for which neighboring values are highly correlated (such as in images). This type of neural network was conceived in 1989 with the appearance of LeNet, a network designed for the recognition of handwritten digits (LeCun et al., 1989). However, they did not become relevant until Krizhevsky, Sutskever and Hinton developed in 2012 the AlexNet, a convolutional neural network which improved its predecessor and won the ImageNet image classification contest (Krizhevsky et al., 2012).

CNNs are networks which follow the same principles as the traditional ANN, but they proceed differently to extract their internal representations. These results are obtained by convolving the input data with a matrix whose weights are learned, usually referred in the literature as kernels, n times, which produce additional outputs usually referred to as filters. As we have mentioned before, the significant difference between the CNN and the ANN lies on the way of operating internally its connections. Instead of performing matrix products and sums of weights on the dense layers, these specific networks perform a convolution. Mathematically speaking, a convolution is an operation between two functions which determines how the shape of one affects on the other. Empirically, the convolution performed in CNNs is defined in equation 2.1, where y is the resulting feature map, x is the input image and w is the kernel applied to extract a specific feature. A visual example on how a 2D convolution is applied in an image is depicted in Figure 2.4.

$$y(t) = (x * w)(t) = \int x(\tau)w(t-\tau)d\tau \tag{2.1}$$

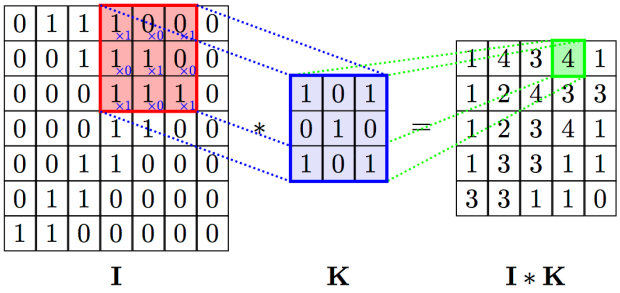


Figure 2.4: Visual representation of a convolution in a CNN

The relevant features of the image depends on the applied kernel properties, such as straight lines, corners or edges. Each convolutional layer is provided with various filters, depending on the neurons that compose it. The result of each neuron is a map of specific particularities of their input. These outputs are subsequently added together, and the result is given an activation function and a bias, as we have seen in the basic procedure. Usually, after each convolution layer, a technique is performed to obtain the most relevant characteristics for the next convolution layer. It is called subsampling. Within it, the most commonly used methods are Max-Pooling and Average-Pooling.

Subsampling consists of applying a kernel to the input and, by means of a convolution, obtaining its maximum or average value for each group of pixels. This way, the image is reduced in proportion to the size of the filter applied.

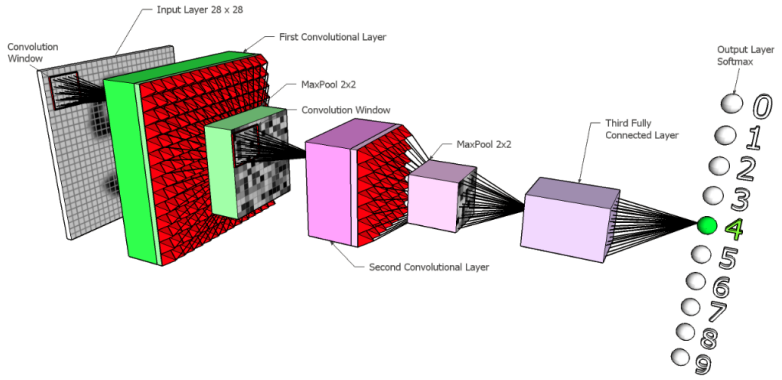


Figure 2.5: Visual representation of a classification process performed by a CNN

Thanks to subsampling, CNN is able to detect high-level particularities of the final image. This fact allows us to differentiate it from other images and, therefore, to classify it in a group—a visual version of this structure defined in the Figure 2.5. CNNs are crucial in the OMR process since we need to detect both staves in a given score and the symbols into each one of them. As it can be interpreted, the convolutional block learns the relevant features of the image to be lately transcribed.

2.3. Brief introduction to Machine Translation techniques

In this section, we focus on briefly reviewing the models used in the Machine Translation research area, as these models constitute a core part of the development of this project. We take this review from a historic perspective, as this project does cover some variety of models that Machine Translation has, from the statistical approaches till the neural ones.

The first NLP approaches that took into account Machine Translation, as they began with the development of chatbots like ELIZA during the 60s, were based on rule-based systems where the translation between two languages process was processed by following a certain set of rules that derived the system's behavior. However, this proposition has a significant problem in scalability terms, as they grow exponentially in complexity as bigger the processed languages semantic rules and structures grow. That is why, over the 80s with the creation of the web, these systems started to coexist with other statistically-based translation systems, which reduced that amount of work that rule design supposed.

2.3.1. Statistical Machine Translation

This approach is based on the family of techniques known as Statistical Machine Translation (SMT) (Koehn, 2009). SMT is a data-driven approach to MT in which several independent models are combined to produce a translation from a text in the source language (SL) into the target language (TL). The basic models in an SMT system are:

- **The translation model**: this model is learned from a parallel corpus, i.e., a set of sequences in the SL with their translation in the TL, and it is used to obtain, given a phrase (i.e. a word n-gram) in the SL, the most probable equivalent phrases in the TL; and
- **The language model**: this model is automatically learned from a TL monolingual corpus, and is used to obtain the probability that a predicted segment belongs to the TL.

In addition to the translation model and the language model, modern SMT approaches integrate additional models, such as a distortion model (to reorder words during translation) or a length penalty. All these models are integrated through a log-linear combination; the weight of each model in this combination is established through a simplex-fashion algorithm that optimizes an automatic quality score on a development set. The combination of all these models allow SMT systems to provide translations that are balanced in translation accuracy and TL fluency.

This approach is currently used in some specific NLP scenarios, specially when tackling languages with few resources or in pre/post-processing tasks. However, these approaches also do not perform as well as other approaches when the processed languages are complex, as they usually require additional information and feature engineering processes to obtain their best performance. Here it is when the neural network-based approaches appear.

2.3.2. Recurrent Neural Networks, Sequence-to-Sequence and Attention

The first neural networks that started to address NLP tasks and Machine Translation processes were the recurrent ones.

Conventional neural networks are capable of solving diverse problems. However, there are situations in which they do not find accurate results if they do not have contextual information. A clear example would be to predict the next value in a numerical series, where it is crucial to take into account the previous outputs to fit the rule. In 1990, Jeffrey L. Elman presented a new architecture to solve this inconvenience that traditional neural networks could not handle: the recurrent neural network referred to as RNN (Elman, 1990).

A Recurrent Neural Network (RNN) is a network whose neurons have an internal loop which allows them to maintain their state over time. This way, neurons not only have information about the immediate moment, but they also contain data about their previous states. The structural representation of a recurrent neuron is shown in Figure 2.6.

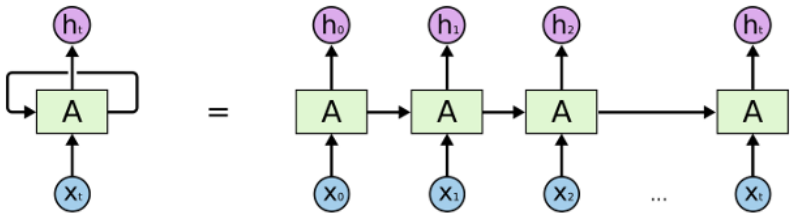


Figure 2.6: Depiction of a recurrent neuron in the Elman model, the first proposition of recurrence in a neural network.

However, this model presented a problem: long-term dependencies. A simple example of this drawback lies in the identification of terms in sentences. There are time series that only need recent past states. For example: "The clouds are in the _", where the term to be predicted found by connecting the two previous words. However, in more complex series such as: "I was born in France, but at the age of twelve I came to Spain ... that is why I speak well _" requires going back to the beginning of the sentence and find out which language the speaker is referring to. In other words, the longer the input sequence, the more convoluted it is to connect the terms of the input to the correct prediction, and therefore the more difficult it is to learn. This problem, in formal terms, is known as the vanishing gradient.

In an in-depth look at this problem, we could make an analysis of the RNN mathematical expressions of their backpropagation process, referred in the literature as the Backpropagation Through Time process (BPTT) to unveil the reason of this notorious inconvenience that RNNs bring in their training process. Given a loss function which computes the sum of all the loss functions L through time T, we could express it as such:

$$L = \sum_{t=1}^{T} L^{(t)} \tag{2.2}$$

Since this loss in a time step t is dependent on the hidden units at all the previous time steps, the gradient is then computed as it follows:

$$\frac{\partial L^{(t)}}{\partial W_{hh}} = \frac{\partial L^{(t)}}{\partial o^{(t)}} \times \frac{\partial o^{(t)}}{\partial h^{(t)}} \times (\sum_{k=1}^{t} \frac{\partial h^{(}t)}{\partial h^{(}k)} \times \frac{\partial h^{(}k)}{\partial W_{hh}}) \tag{2.3}$$

The interesting component of this formula to analyze is the $\frac{\partial h(t)}{\partial h(k)}$ term, which is computed as the multiplication of the adjacent time steps of the current neuron we are computing its backpropagation:

$$\frac{\partial h(t)}{\partial h(k)} = \prod_{i=k+1}^{t} \frac{\partial h(i)}{\partial h(i-1)} \tag{2.4}$$

If we take a look at the multiplicative nature of this factor when computing the gradients of a loss function, we can observe how this problem arises. $\frac{\partial h(t)}{\partial h(k)}$ has $t-k$ multiplications. Therefore, multiplying the weight assigned to this specific recurrent loop, which we denote as w, by itself $t-k$ (the currently elapsed time steps) times results in a factor of $w(t-k)$. As a result, if $|w| < 1$, this factor becomes very small when $t-k$ is large (long-range dependencies). Empirically, this length is considered as 20 tokens. On the other hands, if $|w| > 1$, the factor becomes very large when the long-range dependencies are also extense (gradient exploding). One naive solution to solve this is to ensure that $|w|$ is always 1. However, this goes against the nature of machine learning purposes, as we are heavily limiting weight combinations that could benefit the network performance. There were two practical approaches to solve this issue:

- **Gradient clipping**, which consists on establishing a cut-off value for the gradients which limited their value.
- **Truncating Back Propagation Through Time (TBPTT)**, which consists in limiting the number of time steps that had to be backpropagated after each forward pass. For example, if a sequence is 200 tokens long, we only backpropagate through the most recent 20 time steps.

While these two popular solutions were able to solve the exploding gradient issue, this truncation limited the number of time steps that the gradient can effectively flow back and properly update the weights. Not only to mention that the vanishing gradient problem still existed despite these efforts to solve this training drawbacks. Due to these circumstances, the RNNs did not become relevant until a new model of recurrent neuron emerged: the Long Short-Term Memory (Hochreiter & Schmidhuber, 1997).

LSTM neurons managed to solve this issue thanks to the implementation of an internal memory cell which is controlled by several *gates* which control the value of its activation (known as the input, the output, and the forget gates). The weights of these gates are fine-tuned during the training process, as they have to adapt to the data that conforms the problem to solve. An in-depth understanding of these neurons can be found in Figure 2.7. The first component, and most important one, is the one represented with the $C(t-1)$ and the $C(t)$ on the top of the image. This component is known as the Constant Error Carousel, which ensures a constant error flow, which stores its value indefinitely. By being able of doing so, maintain the error over time, the vanishing gradient is therefore solved. However, this error has to be updated or changed depending on the needs of the problem that is being solved. To do so, two possible influences are then established. The first one, it is the one represented in the f pipeline, which is referred in the literature as the forget gate. This part, consists of a weighted matrix, produced by a multi-layer perceptron which takes the current

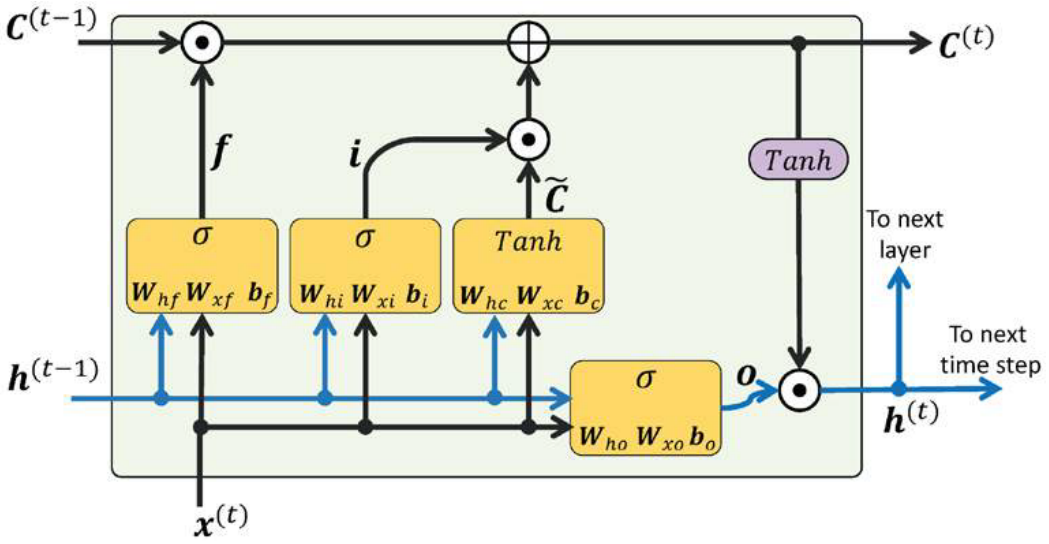


Figure 2.7: Basic depiction of an LSTM cell structure

input and the state of the previous time step, whose values oscillate between zero and one doe to a sigmoid non-linearity that is applied. This matrix computes an element-wise product with the constant error carousel to, if it is required, override or soften the propagated error through the rest of the network. In other words, it determines the influence of the previous time-steps error state to the rest of the network.

Then, we have the input gate, which regulates the influence of the current neuron to the next time steps. This gate is important, as if it were no control at this point, the constant error carousel would have an initial value that only could be decreased by the next time-steps. This way, we produce a proposal influence matrix $\tilde{C}$, whose values oscillate between 1 and -1 due to the hyperbolic tangent application to the outputs of this specific multi-layer perceptron. This influence matrix is then regulated by the input gate, which applies, as the forget gate, an element-wise product to it. Therefore, the obtained result is applied to the constant error carousel with an element-wise addition.

Finally, we have to regulate the influence of our internal state to next time-steps, as we cannot forget that these neurons are recurrent. This is done with the output gate, which computes a matrix with a sigmoid activation which is applied with an element-wise product to the hyperbolic tangent application of the current constant error carousel value to determine our propagated state. This way, we can determine how our state is passed through the network, being 0 if it is required to not take into account this time step or $\sigma(W_{xo}x_t + h_{(t-1)}W_{ho}) \odot tanh(C^{(t)})$ if we consider the internal state to be fully applied.

This way, the network can store and retrieve all the necessary states to optimize its predictions during both short and long periods of time. Thanks to this, the vanishing gradient problem was tackled.

Once understood the relevance of these neurons, we can now describe the implemented model for the translation process, as it is an ensemble of these recurrent layer types.

In Machine Translation, the most popular structure based in neural networks structure is the Sequence-to-Sequence approach (referred as Seq2Seq) (Sutskever et al., 2014).

Its popularity is due to the low coupling of the networks that compose it and its independence from additional information about the relationship between the sequences to be processed. In formal terms, the goal of this model is to estimate the conditional probability of a particular output sequence given an input sequence. It should be noted that the length of these two may differ.

To solve this task, Ilya Sutskever's work (Sutskever et al., 2014) took advantage of the ability of LSTM neurons to store an internal state, represented by a vector of fixed dimensions, with relevant information about the data processed so far. This internal state was also admitted in other LSTM neurons which operated from it as a beginning point. This way, a neuronal model composed of two networks that profited these advantages emerged. The structure of this model is the following:

The first network is an encoder. It processes a sequence element by element and encodes the relevant information within its internal state. This critical piece of the system is known as the context vector. We could understand the context vector as a reduced multi-dimensional representation of the input embedding of the encoder, where the most relevant information of the sentence is described there.

The second network is a decoder. In this case, the system predicts an output sequence by means of the previously produced context vector, which serves as a global information element about the complete input sentence that has to be translated, and the previous outputs. In this structure, we can observe the fundamental role the encoder has to play in this system. If we simply combined the input sentence and the previously predicted tokens, we would limit the system to the information that has seen at the current time step in both the sentence to translate and its current predictions. However, this is not correct at all, as in a real-case scenario of manual translation, we do know all the information about the sentence we have to translate. In other words, we are limiting the system with information that should already know. Thanks to the ability of the LSTM architecture to transfer their internal state activations, we can produce the complete information about the input that the decoder should take into account when producing its translations, without any information restrictions in that part.

This way, the network is able to predict sequences more accurately than traditional systems and, furthermore, it does not require additional information about the relationship between these sequences since this is ascertained thanks to the internal states of the encoder. A graphical example of this network can be found in Figure 2.8

Since its inception, this model has evolved in the machine translation context. Specifically, this evolution has been directed towards the Attention Mechanisms (Luong et al., 2015), which are masking matrix operations that compute the relevance of certain input and output tokens to define the next prediction. In formal terms, attention matrices represent the similarity between the concepts that lay into that matrix. These mechanisms aim to assist the model predictions by enhancing the values of the most relevant information of an input to predict the next token. Attention-based mechanisms have been the key to evolve the described Seq2Seq model into a new level, as they bring the network the ability to not only take into account all the context information generated by the encoder, but to select it depending on their relevance when predicting certain token during a given time step, which grants the network with the possibility of generating better language models and establish precise relationships between the input and the target vocabularies.

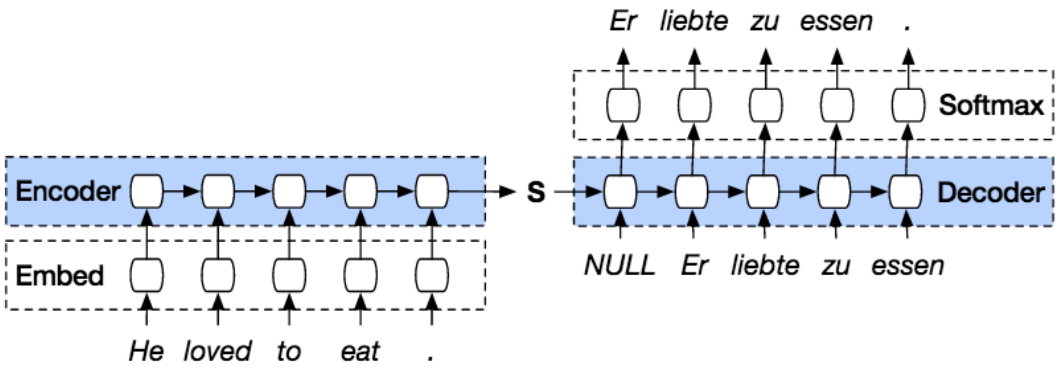


Figure 2.8: Graphic representation of the Sequence-to-sequence neural network implementation for a Machine Translation task between English and German languages.

In fact, the state of the art of Machine Translation, and Deep Learning in general, has evolved towards more sophisticated models completely based on these mechanisms, which brings us to the next presented model architecture in this work.

2.3.3. The Transformer

The Seq2Seq model described above is a reference model in Machine Translation to perform the tasks we are facing. Due to the significant improvements over traditional models the attention-based approaches provide, they have gained significant popularity during the recent years. This has led the research community to develop new systems that take advantage of these elements. This leads us to the second model of interest in this work: the Transformer, which currently represents the state of the art in Natural Language Processing tasks (Brown et al., 2020; Devlin et al., 2019) and has been extended to other Deep Learning-based areas such as Computer Vision (Carion et al., 2020). The Transformer, firstly presented in (Vaswani et al., 2017), is a model that proposes a new approach on computing the encoding and decoding process presented in the previous section. Instead of implementing recurrent neurons to compute these processes, this model presents an alternative formulation based on linear neurons and attention mechanism operations, which are referred to in the literature as the Multi Headed Attention neurons (MHA), that outperformed the classic Seq2Seq systems, among other benefits such as computing benchmarks due to their high parallelization.

A simple description for this mechanism is that the system evaluates the relevance of the sequence elements to describe a specific token by means of computing attention between them. In an in-depth look at the systems formulae, we find how this attention is specifically applied in all layers. Considering an input sequence X and its embedding vectors as $[x_1, \ldots, x_n]$. First, we are going to take a look at an embedding level, x_1 for example. This embedding is computed in three different linear layers to produce the three main weighted vectors the Transformer works with: the query vector (q_1), the key vector (k_1) and the value vector (v_1). Once obtained these vectors and the ones from the rest of the embeddings of the sequence, we then compute the attention mechanism, which results on being a scaled dot product between the query and the key. Formally speaking, the attention score in a specific Transformer head

is computed (at a vector level) as it follows:

$$\text{Score}(q_1) = \text{softmax}(\frac{q_1 \cdot K^T}{\sqrt{d_k}}) \tag{2.5}$$

Where q_1 is the query vector representation of the first embedding, K is the vector composed by the produced key vectors by row (being the first row k_1, the second one k_2, etc) and d_k is the dimensionality of the K vector. As we can observe, we obtain a weighted vector which represents the relevance of the elements of K regarding the q_1 embedding. Finally, we compute the product between the score and the value vector of this embedding v_1, which we refer as z_1, which represents the original embedding with the attention applied.

Of course, as we are dealing with a vector linear application, this calculation can be represented as a matrix operation, so we can generalize the formula as:

$$Z = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_k}})V \tag{2.6}$$

This computation is done n times, which is known as the attention heads or multi-headed attention mechanism, which is a mechanism conceived to make the Transformer to "tighten its bets" and avoid the dependency problem on the randomly-generated weights in the Q, K, V linear layers. These heads are then concatenated and passed through a feed-forward module to sum up all the data into one output. A graphic implementation of this specific layer can be found in 2.10. Once understood this concept, then we can specify the two situations that can be seen in the Transformer architecture. The first one (found in the middle point of the decoder) is the MHA pure concept, where the Q matrix consists of the embedding representations of the last encoder layer, and the K and the V matrices are the outputs of the decoder layer. The second situation is the one referred in the literature as the self-attention mechanism, where the Q, the K, and V matrices are computed from the same input embeddings. This concept is key, as the Transformer does not only deal with the relevant parts of the encoder output for the decoder to produce optimal embeddings in the translation process, but it also generates better embeddings by seeking both the grammatical (in the first layers) and semantic (in the latter ones) relationship between the tokens presented in the input sequence.

This described process is applied in both the encoding and the decoding steps of the translation mechanism. By replacing the recurrent layers with the MHA layers, the system, as reported in (Vaswani et al., 2017), gains significant performance without losing the benefits that the state of-the-art recurrent neurons, such as Long Short-Term Memory ones, had. It is also remarkable the improvement of the training benchmarks the Transformer has in comparison to the recurrent models, as it is a model that is easier to parallelize, as it is not bounded to the concept of sequentiality that recurrent networks have and, as it is based in lineal applications, most of its steps can be summarized in matrix operations. Even though, this lack of sequentiality is an issue that has to be addressed, as in our case it is mandatory to have the model know a certain order between sequence tokens. The original paper also proposes an elegant token coding named Positional Encoding, which are mathematical functions (based on the sine and cosine properties) that produce singular results depending on the position of the token in the sequence. Thanks to this process, which is also accompanied by masking operations to avoid position foreseeing in the decoding step, the Transformer is able

to process text sequences without losing positional information, which makes it a powerful model to use in Machine Translation applications.

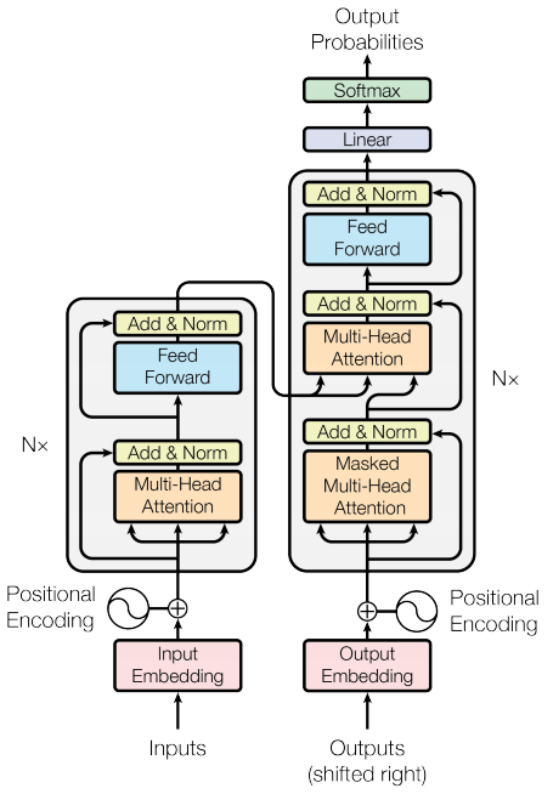


Figure 2.9: Transformer neural architecture, taken from the original paper (Vaswani et al., 2017)

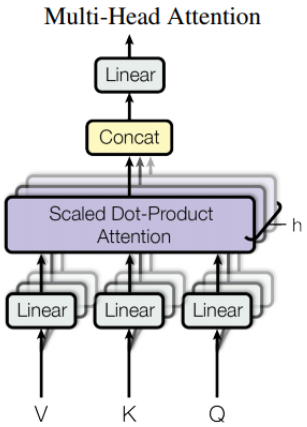


Figure 2.10: Graphic description of a Multi Headed Attention layer implementation

Once described the state of the art that is present in this work and will be applied as new formulation in the OMR field, we then have to establish our main goals an determine how we are going to organise this work to conduct our research.

3. Materials and methods

In this chapter, we present all the tools and resources used to design, create and evaluate the proposed complete OMR pipeline. We review all the used tools and systems for the experimentation and describe how the presented data, which is detailed and introduced as it is needed in the following chapters, has been curated, labelled and processed to be usable in our experimental scenario.

3.1. Tools

In this section, we comment and describe the main tools and libraries used for this project. We subdivide this section in the areas each tool has been covered during the development of this work.

3.1.1. Statistical system implementation

In the previous chapter, we mention the Statistical Machine Translation techniques, as we use them during the implementation of the proposed systems in Chapter 4. For this approach, we used the open source Moses toolkit for SMT (Koehn et al., 2007). This system was conceived in 2007 and is currently one of the most well-known SMT solutions for experimental and research scenarios, as it provides a wide variety of tools to customize and fine-tune a data-driven translation system. As they describe in their web page, we can find the following features in the tool:

- Two types of translation models: phrase-based and tree-based
- Support for factored translation models, which enable the integration of linguistic and other information at a word level
- The tool allows the decoding of confusion networks and word lattices, which enable the easy integration with ambiguous upstream tools, such as automatic speech recognizers or morphological analyzers.
- The integration of an experiment management system, which makes Moses an easy-to-use and productive tool for research scenarios.

We consider this tool is suited for our task, as the statistical translation between musical languages is a topic that has not been explored yet, so it is required a flexible and usable tool to handle the fine-tuning of several models with varying configurations.

3.1.2. Neural Networks implementation

In this section, we present the main libraries used to develop the neural approaches proposed for the recognition systems and machine translation approaches. All the presented libraries have been run on the Python environment, which is an interpreted programming language whose main focus is to bring simplicity and legibility to the produced code (on the exchange of performance). However, most of the tools we are using are compiled C++ binaries (focused on speed execution) that can be handled and used in this simpler environment. This feature brings us a fundamental advantage: we can produce simple legible code without losing performance on the system's execution (as all the calls and instructions executed in the library are handled in an optimized binary instructions set).

3.1.2.1. Tensorflow

Tensorflow (Abadi et al., 2016) is an open-source library for numerical computing using data flow graphs. These data are organized into multidimensional vectors known as tensors. These elements circulate through mathematical operations and follow a connected network structure. Tensorflow was developed and released by Google Brain in 2015 under an Apache 2.0 license. Its core is programmed in C++. However, it has interfaces for its use in high-level languages such as Python, Goes and Java. It is possible to run Tensorflow on both the Central Processing Unit (CPU) and the Graphical Processing Unit (GPU). In our case, we will use the second option.

Currently, the library is a popular platform for building and training neural networks and has become one of the reference technologies for the development of machine learning algorithms. Its popularity is due to its philosophy, which considers that any mathematical operation, regardless of its complexity, can be expressed as a set of primitive functions and operations. This perspective has excellent advantages for the creation of deep learning models. It eases the derivative calculation of each individual operation and the application of a chain rule, which is useful for computing loss functions based on gradient descent methods, an essential technique for the development of current deep learning programs.

3.1.2.2. Keras

Keras[1] is a neural network development framework conceived by François Chollet in 2015. Keras aims to be a middleware between the developer and another supported ML programming libraries. Keras enables prototyping and development of neural models for deep learning easily. In other words, it adds an abstraction layer to ease the implementation process of, for example, a neural network in Tensorflow. The user only uses its simplified language, which is then translated into the targeted library implementation. Keras has support for Tensorflow, Theano and Computational Network Toolkit. His philosophy is based on moving from the idea to the result with as little delay as possible, as they believe this is the key to proper research. It is also worth noting that, since version 2.0, Tensorflow currently incorporates Keras as a main implementation technology, so it is usual to see this tool referred as TF/Keras, as they come in a single package for the Python environment. In this project, we work mainly with Keras.

[1] https://keras.io/

3.2. Computing resources specifications

In this work, we dive into time benchmark comparison between models performance. Because of this, it is necessary to detail the technical specifications, for the sake of completeness, of the computers that have been used during the training of the proposed for this project. The computing resources used for this project are detailed in Table 3.1

Table 3.1: Computing resources table with relevant specifications of the components used in each server.

Computing Server 1 (SMT Models)	
Component	**Component specification**
Processor	AMD Opteron(tm) 6128, 8 cores
GPU	-
RAM	64 GB DDR4
Computing Server 2 (Neural models)	
Component	**Component specification**
Processor	Intel® Xeon® E3-1230 v5
GPU	NVIDIA GTX 1080, 8GB VRAM
RAM	32 GB DDR4

3.3. Data processing and curation

A relevant part of this work is the retrieval and labeling of the data used for this work. In order to prepare consistent and representative musical data sets, we used the Music Recognition, Encoding, and Transcription tool (referred in the literature as MuRET) (Rizo, Calvo-Zaragoza, & Iñesta, 2018), which is an online tool devised to cover all transcription phases of a music document digitization, from the manuscript source to the encoded digital content. MuRET is designed as a technology-focused research tool, allowing different processing approaches to be used and producing the expected transcribed contents in standard encodings and data for the study of the transcription process itself.

The MuRET pipeline, which is still under research and development, consists in a two-step labeling process. The first one is the Document Analysis step, where all the significant regions of a music score are labelled in bounding boxes (i.e, musical staffs, lyrics or the title of the score). A representative image of how this interface is presented can be observed in Figure 3.1. Once labelled these significant regions with a bounding box, we proceed to the second process, which is the Agnostic Transcription step. This part can be approached by different means. In the manual process, the user is able to both label the notes with their approximate bounding boxes or, if it has technical resources like graphic tablets, can outline the shape of the labelled note, which the tool processes automatically to infer the

most suitable bounding box to cover the selected symbol. Once done this selection, the user must specify which is the shape and the position of the note in the staff. In other words, the user gives the agnostic encoding of the specified note. As we are dealing with a non-standard musical language which may be unknown to the user, MuRET enables, as it can be seen in Figure 3.2, a graphical tool to represent the selected bounding box as a note in the staff (it delivers a graphics representation of all the possible musical notes which then are encoded into their agnostic language respective).

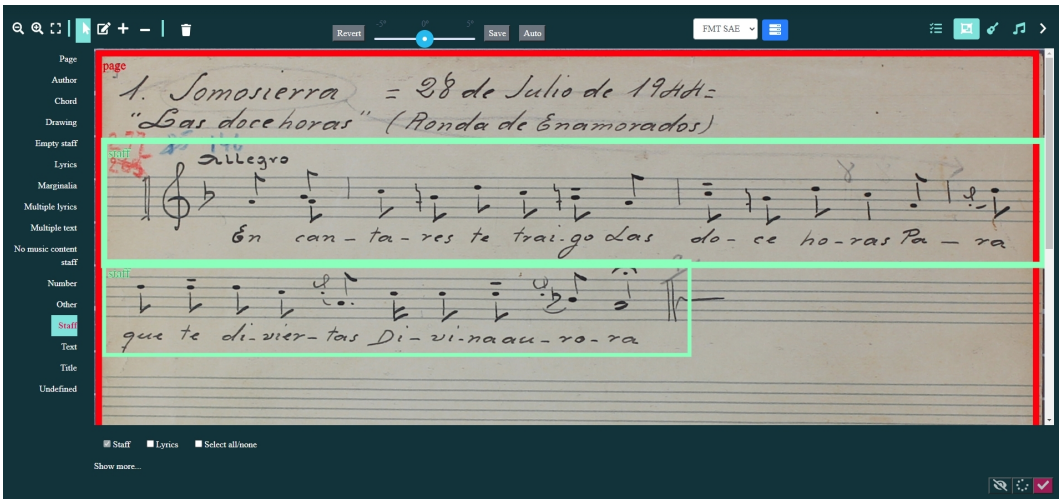


Figure 3.1: User interface visualization of the Document Analysis labeling step in the MuRET tool

Once obtained the graphics-based representation of the music score, the user is required to produce the same representation in a semantic encoding. This is done with a tool MuRET provides below the graphical encoding interface, where the user, who we understand to have knowledge of digital musicology languages, has to type the semantic representation of the music staff that is being transcribed.

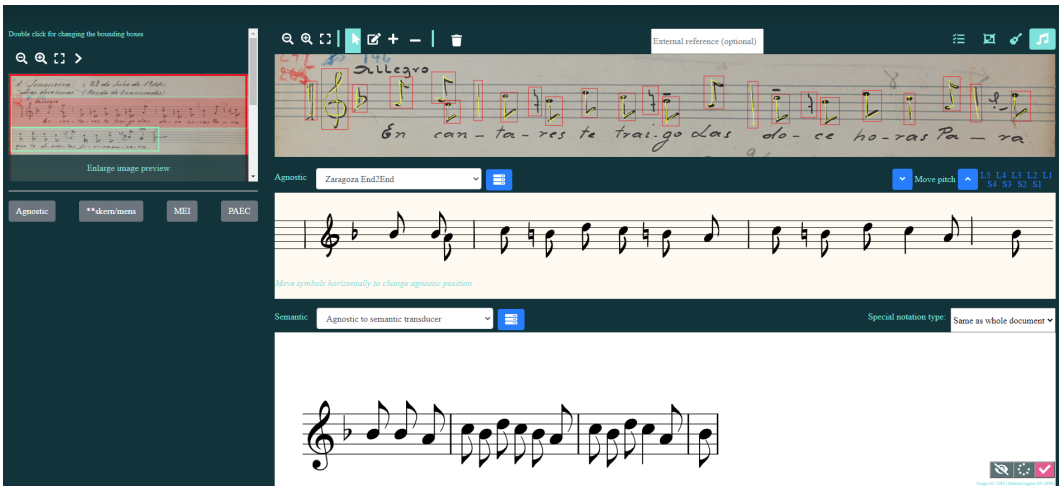


Figure 3.2: User interface visualization of the Transcription labeling step in the MuRET tool

Of course, we have described the manual tagging process (which is the most inefficient one) in order to have an understanding of the steps required by the tool to store the data for each musical score. However, MuRET provides state-of-the-art OMR tools which ease the work the user has to perform during the musical transcription and labeling process. For example, in the Document Analysis step it can be found a Selectional Auto Encoder (Castellanos et al., 2020) region detector, or the mentioned end to end music transcription systems for the next steps. However, it must be noted that, when beginning the labeling process of a new corpus, the described manual workflow is required until there is enough data to train the automatic systems to aid in the process, which is something that had to be dealt with in the case of some presented corpora during this work.

Once finished the labeling process of a corpus, one final step is required in order to obtain a usable data set for our project: exportation. Luckily, MuRET takes into account the needs of some users, specifically researchers, and integrates an automatic tool for data exportation in easy-to-read documents. The tool is able to give the labeled corpora in a JavaScript Object Notation (JSON) document, which easily to read and retrieve with the Python built-in JSON parser library. Specifically, the JSON files used during this process had the following structure:

- regions: An array object that contains all the labeled regions in the music score document. For each region, we can find the following data:
  - image_name: File name of the image where the region belongs to. This item is useful for the image recognition process, as it allows a fast document retrieval when processing the data.
  - region_id: Identification key of the given region.
  - bounding_box: Bounding box limits where the labeled item is contained. It comes in an anchor point format: $[x_f, x_o, y_f, y_o]$, where the sub indexes of o and f refer to the origin point or the final one.
  - region_type: Label given to the labeled region.
  - agnostic: If the region is labeled as a "staff", this data contains the agnostic encoded sequence of the represented region. If not, it contains a null value.
  - semantic: If the region is labeled as a "staff", this data contains the standard semantic encoded sequence of the represented region, the encoding type of this specific data is discussed in the following chapter. If not, it contains a null value.

As we can observe from the following list, the data retrieval to train our desired systems is rather trivial. The algorithm to do this retrieval can be described as it follows:

1. Create an empty image vector.
2. Create an empty agnostic sequences vector.
3. Create an empty semantic sequences vector.
4. For all the regions present in the document:

a) Retrieve the image from the **image_name** parameter and crop it with the given bounding box.

b) Parse the agnostic sequence with regex expressions to obtain a space-separated vector and append it to the agnostic sequences vector.

c) Parse the semantic sequence with regex expressions to obtain a space-separated vector and append it to the semantic sequences vector.

5. Return the produced vectors.

Once described and detailed the labeling and the information retrieval processes, we are able to start our experimentation phase, which is described in the following chapters.

4. Applying Automatic Translation for Optical Music Recognition Encoding Step

In this chapter, we focus on the analysis and evaluation of performing Machine Translation, referred to as MT through the rest of this report, techniques in the musical encoding's domain.

The chapter is organized as it follows: we first formalize and describe in Sect. 4.1 the task of applying Machine Translation to music encoding as well as we make an in-depth analysis of the used music codifications in this work. Then, in Sect. 4.2, we propose and describe several approaches from the Machine Translation area to be evaluated, from the statistical options to the state-of-the-art systems on the topic. In Sect. 4.3, an experimental setup is established where the corpora used in these experiments is presented, and describe the specific challenges and corner-cases that involve using this kind of automatic translation systems between music encoding representations. In Sect. 4.4, results are presented and the performance of the proposed models is discussed in-depth. Finally, in Sect. 4.5, we conclude this work and explain our final thoughts about the convenience of using these MT techniques for the encoding step in the OMR pipeline.

4.1. Theoretical preamble

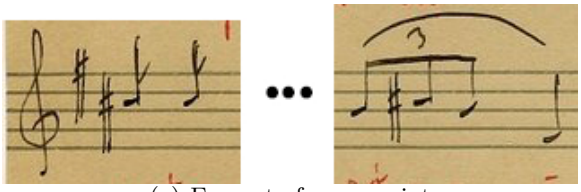
Several approaches to obtain a digital score file from an image have been proposed so far (Calvo-Zaragoza et al., 2020). Discarding any image processing and document analysis considerations, and focusing on how the final standard notation format is obtained, two main approaches can be found: an end-to-end, image-to-semantic approach, and systems that yield an intermediate representation that needs to be converted into a standard encoding such as MEI (Hankinson et al., 2011), MusicXML (Good & Actor, 2003) or ***kern* (Huron, 1997). As we have seen in this report introduction, the most common way to perform OMR is to produce this intermediate representation that tries to explain the content in the score in terms of graphical elements. Meaningful examples of this category are that of Pacha et al. (Pacha et al., 2019) where a notation graph is generated from a set of primitives, such as note heads, stems, beams, accidentals. Also, in (Calvo-Zaragoza & Rizo, 2018), a simpler representation denoted as *agnostic* is used that consists of musical symbol sequences instead of considering their constitutive primitives. This approach has been shown to be the most robust one. However, as we have mentioned before, this graphics-based encoding needs to be converted into a standard music notation format.

This work focuses on the second phase of this approach: the translation between the agnostic encoding of single monodic staves into a standard semantic format, which is an extension of the ***kern* format described and discussed below in Sect. 4.1.1.

Specifically, a sequence of symbols is extracted from the segmented image (Fig. 4.1(a)), using any method. These are agnostic symbols characterized as a graphical glyph, such

as accidental, rest, clef, or note, and its vertical position in the staff (see Figure 4.1(b)), and do not have a direct unambiguous semantic meaning. For instance, in the excerpt from Figure 4.1(a), the second sharp is neither assigned the function of key signature, nor alteration of the next eighth note in the staff. They are just a sequence of two symbols of pitch alteration detected in a particular vertical position, in this case the fifth line and the third space. The digit '3' found in the second part of the example image is labeled as a digit 3, and its actual meaning (maybe a numerator in a fractional time signature, or a triplet number in a tuplet) will be assigned later when this representation is translated to the semantic encoding. We refer the reader to detailed information about this agnostic encoding format in (Calvo-Zaragoza & Rizo, 2018).

Therefore, the task under analysis is the translation of this sequence of graphical symbols into a semantic encoding expressed as a standard music representation format. The ***kern* format is used in the example in Fig. 4.1(c) and an extension of it will be considered as the output language for the approaches described below in 2.3.1 and 4.2.1.



(a) Excerpt of manuscript

```
clef.G2:L2, accidental.sharp:L5, accidental.sharp:S3,
note.eighth_up:S3, note.eighth_up:S3, ...
slur.start:S5, note.beamedRight1_up:L3, accidental.sharp:S3,
digit.3:S5, note.beamedBoth1_up:S3, note.beamedLeft1_up:L3,
slur.end:S5, note.quarter_up:S1
```

(b) Agnostic encoding

```
**kern
*clefG2
*k[f#]
8cc#/
8cc#/
...
(12bL/
12cc/#
12bJ/)
4f#/
*-
```

(c) Semantic encoding (note how the sequence of two accidentals after the clef is actually a G Major key and an accidental of the first eighth note).

Figure 4.1: Agnostic to semantic encoding example

From a formal perspective, we can define the task to be performed as the following: given

a sequence of agnostic symbols, $\mathbf{s} \in \Sigma_a^*$, the translation step f_t can be expressed as seeking the sequence $\widehat{\mathbf{t}}$, such that

$$\widehat{\mathbf{t}} = \arg\max_{\mathbf{t} \in \Sigma_s^*} P(\mathbf{t} \mid \mathbf{s}). \tag{4.1}$$

where $\widehat{\mathbf{t}}$ is the predicted semantic sequence, $\mathbf{t}$ is the targeted semantic sequence ($\mathbf{t} \in \Sigma_t^*$) and $\mathbf{P}(\mathbf{t} \mid \mathbf{s})$ is the probability of predicting the desired sequence given the mentioned input.

4.1.1. Output semantic encoding

An important determinant of any translation effort is to find a semantic output encoding suitable for being obtained by the translation system. To achieve this goal, it is necessary to analyze which encoding is most beneficial in terms of exportability and compatibility with the most used musicology tools.

The first options that may be considered are the most extended semantic encodings in libraries and musicology contexts: MEI (Hankinson et al., 2011) and MusicXML (Good & Actor, 2003), which represent a music score components and metadata in an XML-based markup attribute encoding. These can be understood as analogous markup-based encoding languages such as TEI ("TEI P5: Guidelines for Electronic Text Encoding and Interchange", 2007) in the text recognition context. Despite being extended codifications, these semantic representations present a significant drawback when considering their use in a machine translation system, since they are highly verbose. That is, the amount of tokens required to represent a musical symbol is larger than in the agnostic encoding case. This also means that the target language vocabulary would be greater than the input one, even in small music excerpts. This is not convenient, as in this situation word alignment between codifications is harder to perform by machine learning systems, and it would result in a more erratic predictive model, as well as it thereby makes the automatic encoding task unnecessarily complicated.

The encoding of choice in this work is a semantic language known as Humdrum **kern. This is a robust and widely-used semantic encoding for many musicological projects. Its benefits for our purpose lie in a simple vocabulary, a sequential-based format, and in its compatibility with dedicated musicology software like Verovio Humdrum Viewer Sapp (2017), which brings the possibility of automatically converting into other formats, which is its most celebrated feature. An example of the convenience of this format over other XML-based ones, like MEI, is shown in Fig. 4.2.

Despite that, Humdrum **kern has some issues representing specific music symbols present in our corpora. The most relevant inconvenience of raw **kern is due to the analytical purpose of Humdrum, focused on encoding music content rather than visual issues. Multi-measure rests are encoded in **kern as a list of standard measure rests, which sometimes makes the target sequences to be rather long. This encoding of multi-measure rests in **kern, in an OMR context, conflicts with the CTC loss function, which will be used and described during the next chapter, since they represent a long temporal sequence to be predicted within a few amounts of image frames. As in the end of this report, the Machine Translation approaches will be compared with a baseline that directly outputs Humdrum **kern, we need to find a way or an extension that this baseline could produce a competent output. This seek of an

extension also benefits the translation process in the short term, as we reduce the size of the output sequences.

In previous research and experiments carried out on this subject (Ríos-Vila et al., 2020), a semantic encoding based on Humdrum ***kern* was proposed. This extension is known as ***kern**, which is able to be directly converted and reverted from/to ***kern* with trivial sequence parsing operations. As we are dealing here just with monodic sequences, this extension removes all voice managing, bar counting, and layout commands and filters. Nevertheless, they may be included in the system in case of having access to a big enough corpus to train the model. In ***kern**, the multi-measure rests, which have been remarked as the conflicting symbols in the recognition process, have been encoded as 'rr<N>', where <N> stands for the number above the rest symbol. For instance, a multi-measure rest spanning 14 measures would be encoded as 'rr14'. In other words, ***kern** transforms a list encoding into a single token which, in exchange for a slightly larger vocabulary, significantly shortens the output sequences. As some of our corpora were utilized in that previous work and the benefits of the original encoding remain unchanged, we decided to use this format as the output semantic encoding for our experimentation.

4.2. Proposed models

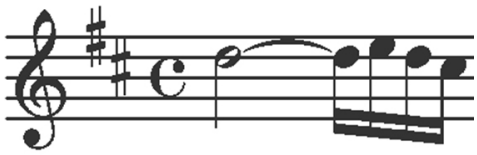
In this section, we comment and discuss the different approaches taken in this work to perform the automatic translation between the agnostic encoding and ***kern**. As we have explained in the introduction, a usual approach in most commercial systems to convert from agnostic encoding to semantic encoding is based on a rule-based translation system. This has been proved to be a challenging task in complex scores (Byrd & Simonsen, 2015; Hajic & Pecina, 2017). This approach also has significant issues in both extrapolability for different notation types, and scalability terms, as it requires the design of new rules or systems when the source and/or the target encoding vary. This is hardly maintainable when the languages reach a significant size. This situation leaves us to look for more sophisticated models in the Machine Translation area, which are described during the next sections. We refer the reader to Chapter 2 to find information about the described models above.

4.2.0.1. SMT

The first implemented model to tackle this process, was the Statistical Machine Translation one. In this case, the well-known Moses toolkit was used, as we have described in the previous chapter, and we computed a Minimum Error Rate Training (MERT) Och (2003) as the algorithm for tuning the models in the log-linear combination.

The optimal configuration for the task was determined by running a grid-search over a series of parameters on the FMT corpus, as this is a medium-sized data set that provides a good balance between the amount of data and the computational cost of running an exhaustive evaluation. As a result of this evaluation, the order of the language model was set to 7 and the maximum size of the phrases in the translation model was set to 5.

However, it is worth noting that the context available during translation is determined by the size of the phrases and the order of the language model. This is a clear disadvantage in front of neural approaches that deal with context in more sophisticated ways (see Sect. 4.2.1).



(a) Example music excerpt

```
... <music> ↵ <body> ↵ <mdiv> ↵ <score> ↵
<scoreDef xml:id="scoredef-0000000430793170" key.sig="2s" meter.sym="common">
<scoreDef xml:id="scoredef-0000000430793170" key.sig="2s" meter.sym="common">
<staffGrp xml:id="staffgrp-0000000321535565">
<staffGrp xml:id="staffgrp-0000000321535565">
<staffDef xml:id="staffdef-0000000979385103" clef.shape="G"
 clef.line="2" n="1" lines="5" />
</staffGrp> ↵ </scoreDef> ↵ <section xml:id="section-0000002102168953"> ↵
<measure xml:id="measure-0000000817881159" right="single">
<staff xml:id="staff-0000000752632627" n="1">
<layer xml:id="layer-0000001525105800" n="1">
<note xml:id="note-0000000088370008" dur="2" oct="5" pname="d" tie="i" />
<beam xml:id="beam-0000000838622227">
<note xml:id="note-0000001323524379" dur="16" oct="5" pname="d" tie="t" />
<note xml:id="note-0000000788593928" dur="16" oct="5" pname="e" />
<note xml:id="note-0000001776562259" dur="16" oct="5" pname="d" />
<note xml:id="note-0000000069259125" dur="16" oct="5" pname="c" />
</beam> ...
```

(b) MEI representation of the music excerpt ('↵' represents the end-of-line character.)

```
**skern ↵ *clefG2 ↵ *k[f#c#] ↵ *met(C) ↵2dd ↵ 16dd ↵ 16ee ↵ 16dd ↵ 16cc#
```

(c) **kern representation of the music excerpt ('↵' represents the end-of-line character.)

Figure 4.2: Example of MEI and **kern representations of a simple music excerpt, showcasing the different verbosity between formats.

We therefore hypothesize that, when translating from the agnostic encoding of a music work including a clef different from the most repeated one, or having a key signature with a number of flats or sharps, SMT methods may lack information to correctly handling the pitch of notes that are not near the beginning of the staff. In order to confirm this hypothesis, we have created a modified version of the agnostic representation, named *contextual agnostic*. This new notation extends the standard agnostic notation by concatenating to each symbol the clef and key signature context in which they are found. Clef is encoded with a suffix equal to the agnostic clef sign. The key signature is encoded as the number of contiguous sharps or flats it is made of (see Figure 4.3).

```
clef.G:L2, accidental.flat:L3, accidental.flat:S4, accidental.flat:S2,
digit.2:L4, digit.4:L2, note.eighth_down@clef.G:L2@3b:L3, verticalLine:L1,
note.eighth_up@clef.G:L2@3b:L2, note.sixteenth_up@clef.G:L2@3b:L2,
note.sixteenth_up@clef.G:L2@3b:L2, note.eighth_up@clef.G:L2@3b:S2,
note.eighth_up@clef.G:L2@3b:S2, verticalLine:L1, note.quarter_down@clef.G:L2@3b:L3,
note.eighth_down@clef.G:L2@3b:L3, note.eighth_down@clef.G:L2@3b:S3, verticalLine:L1,
note.eighth_down@clef.G:L2@3b:L4, note.eighth_down@clef.G:L2@3b:S3,
note.eighth_down@clef.G:L2@3b:L4, note.eighth_down@clef.G:L2@3b:S3
```

Figure 4.3: Agnostic encoding including context

.

4.2.1. Neural approaches

4.2.1.1. Sequence-to-Sequence with Attention Mechanisms

The first neural approach used in this work is based on the Sequence to Sequence (Seq2Seq) system we described before.

In particular, we resort to the "Global Attention" strategy proposed by Luong et al. (Luong et al., 2015), with a scoring method given by the scalar product between the encoder and the decoder outputs. The produced attention matrix in this method acts as a weighting element in order to select which tokens are relevant to produce the next token in the targeted sequence. A representation of the implemented model in this work is depicted in Figure 4.4.

In its current implementation, hereby referred to as the Seq2Seq-Attn approach, we implemented a basic encoder/decoder recurrent architecture (see fig 4.4 for graphical details) with long short-term memory layers. The optimal configuration of the network depends on the word embedding size and the depth of the recurrent layers, where we try to generate narrow vectors, always smaller than the size of the vocabularies used, that group the necessary information to establish relationships between the different tokens that compose the network entries. After performing the best subset selection process to determine these parameters, we found that the network generally got its best performance with a word embedding of 64 and a recurrent layer depth of 128 units. The weights of this network have been fine-tuned with a Categorical Cross-Entropy loss function. A description of this implementation is shown in Figure 4.5

4.2.1.2. The Transformer

In this work, we implemented a simple version of the Transformer model, as the most refined versions require large amounts of annotated data, which were not available for this work. An image representation of the proposed architecture is described in Figure 2.9. Its specific implementation is based on the architecture defined in the original work and depicted in Figure 2.9. However, in this work we produce a simpler model in number of layers and embedding sizes, as the reference implementation was designed to handle languages with a bigger and more complex vocabulary. As we did with the Seq2Seq-Attn approach, we found the optimal configuration parameters for the network with a best subset selection process. The layer implementation of this model is described in Figure 4.6.

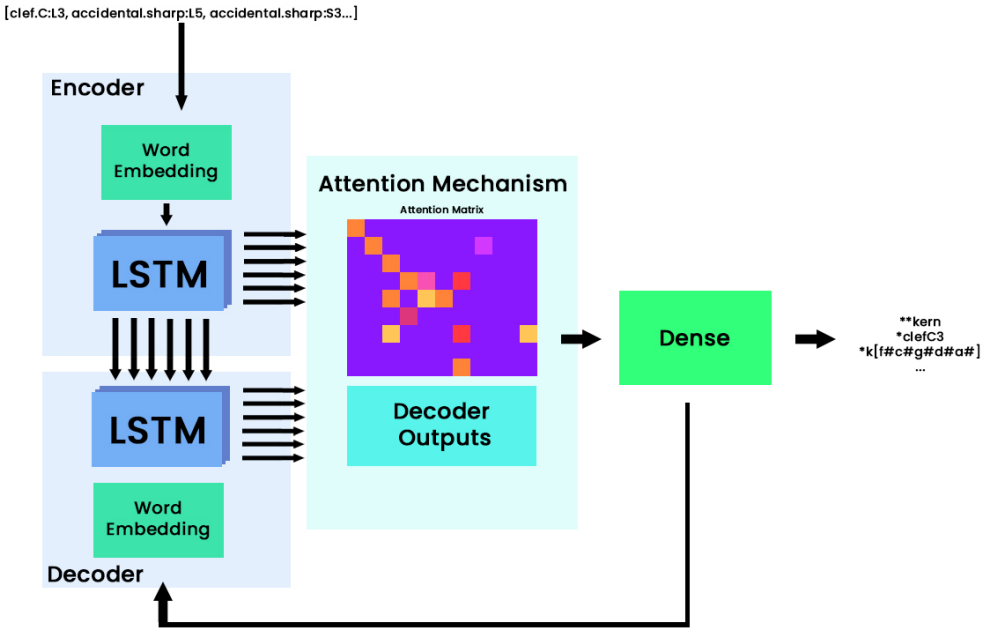


Figure 4.4: Proposed neural architecture for the Seq2Seq with Attention mechanisms model. In this schema, the LSTM (Long short-term memory layers) boxes represent the recurrent layers implemented in the encoding and decoding steps and the Dense boxes represents the last linear layer which classifies the next token in the target sequence. The output of this layer is then propagated to the decoder in order to provide this new information to predict the next target sequence token.

Input $(l_0 \times \Sigma_a)$
Embedding(64)
LSTM(128)
Input$(l_1 \times \Sigma_s)$
LSTM(128)
GlobalAttention()
Dense(Σ_s)
Softmax()

Figure 4.5: Specific implementation schema of the Seq2Seq-Attn approach, where l represents the sample length, Σ_a the size of the agnostic vocabulary, and Σ_s is the size of the semantic vocabulary.

Input ($l_0 \times \Sigma_a$)
Embedding(64)
PositionalEncoding(l)
MHA(64,8)
LayerNormalization()
Dense(128)
ReLU()
Dense(64)
LayerNormalization()
Input ($l_1 \times \Sigma_s$)
Embedding(64)
PositionalEncoding(l)
MHA(64,8)
LayerNormalization()
MHA(64,8)
LayerNormalization()
Dense(128)
ReLU()
Dense(64)
LayerNormalization()
Dense(Σ_s)
Softmax()

Figure 4.6: Specific implementation schema of the Transformer model, where MHA represents the Multi Headed Attention layers, which take a first parameter as their size and a second parameter as the number of heads that the attention matrix has to be split in, l represents the sample length, Σ_a the size of the agnostic vocabulary, even of not being a one-hot encoded input, and Σ_s is the size of the semantic vocabulary.

4.3. Experimental setup

In this section, we present the experimental framework established to carry out our experiment, as well as all the data processing and curation to prepare the described corpora to be used in the proposed models. Once done this description, we also specify the evaluation methodology and the used metrics to compare the performance of the three presented approaches.

4.3.1. Corpora

In order to understand the behavior of the different translation approaches, three different datasets of dissimilar nature have been used. In all cases, they are made of sets of monodic staves, either in mensural or modern notation. The same dictionary of agnostic symbols is used for both notation types. However, the target semantic encoding is different depending on that notation type: in the case of mensural notation ***mens* (Rizo, Pascual-León, & Sapp, 2018), a variant of standard ***kern*, is used. Modern notation is encoded using ***kern**.

The first corpus, named *Zaragoza*, is made of a compilation of 17th and 18th century manuscripts from the 'Cathedral of Our Lady of the Pillar' in Zaragoza (Spain).[1] This dataset is an evolution of the dataset 'Zaragoza' created manually and introduced in (Calvo-Zaragoza et al., 2016) and refined using the MuRET tool (Rizo, Calvo-Zaragoza, & Iñesta, 2018). A complete sample is shown in the appendix Figure A.1.

The second corpus, *Camera-PrIMuS*, is described in (Calvo-Zaragoza & Rizo, 2018). This synthetic dataset was obtained from the RISM collection (Keil & Ward, 2017) and automatically rendered to images, and transcoded to agnostic and semantic notation. The rendering process introduces a variety of image distortion operators in order to produce samples close to real scanned or photographed printed scores. The original version of the corpus contains 87,678 samples, each one containing just one staff. In this work, samples containing mensural notation have been removed from this corpus, in order to have one notation type per corpus. Thus, the filtered dataset actually contains 87,316 staves (see a complete example in appendix Figure A.2).

The third corpus, named FMT, is introduced in this work. It is a collection of four groups of handwritten scoresheets of popular Spanish songs taken from the 'Fondo de Música Tradicional IMF-CSIC'.[2] It is a set of melodies transcribed by musicologists between 1944 and 1960. All music sheets contain monodic staves written in modern notation. The agnostic encodings have been obtained by the same procedure applied in Camera-PrIMuS. See Figure A.3 for a sample of this corpus.

Some music sheets in FMT have a particular feature: in order to save space for the actual notes in a staff, the clef and key signature in some of these manuscripts were written only at the beginning of the piece, as opposed to standard western common practice. This means that in this case most of the staves in a piece (except for the first) have no clef and key signature (see an example in appendix Figure A.3). This was a common practice in the middle XX century period for some sorts of printed music, at least in Spain, such as music for marching bands. This was due primarily to the fact that these manuscripts were often written in small pieces of paper, to make them tractable in different performance situations. Thus, musicians had to learn to 'remember the current key' from staff to staff. Since in this experimental setup input instances are isolated staves extracted from those manuscripts, we should expect models' performance on these samples to degrade due to this issue.

For the purpose of analyzing translation errors, the FMT corpus has been divided into two datasets, FMT-M, containing those pieces that have the clef and key signature correctly notated, and FMT-C, containing those pieces with missing key signature and clefs at the beginning of all staves except for the one at the top of each piece.

Table 4.1 presents the overall statistics of these corpora in terms of instances and symbols. Table 4.2 presents statistics about semantic content of the corpora, including some figures about clefs and key signatures. In particular, the figures about staves with key signature reports the percentage of staves that are part of a piece with an explicitly written key signature. The average ratio of notes altered by key reports the percentage of notes in those staves whose pitch is affected by the key signature.

It is worth noting that the size of the contextual agnostic vocabulary can be, at the most, 15×7 times the non-contextual vocabulary size, as there are 15 possible different key sig-

[1] RISM Code 'E-Zac' at https://rism.info/
[2] https://musicatradicional.eu/es/home

natures and up to 7 different clefs. Actually, in our corpora, the largest increase in the vocabulary size, by factor of around ×16, occurs in the Camera-PrIMuS corpus. This is a potential challenge for statistical models when the size of the corpus is relatively small compared to the vocabulary size, which is the case for the FMT and Zaragoza corpora.

Table 4.1: Characterisation of datasets

	Zaragoza	**FMT**	**Camera-PrIMuS**
Number of staves	140	872	87,316
Mean length agnostic	42	19	27
Mean length semantic	36	19	25
Agnostic/semantic length ratio	0.9	0.97	0.92
Agnostic vocabulary size	200	266	862
Contextual agnostic vocabulary size	506	432	13,721
Semantic vocabulary size	421	206	1,421
Ratio of unknown agnostic symbols(*)	0,09	0,13	0,00
Ratio of unknown contextual agnostic symbols(*)	0,37	0,28	0,02

(*) in test sets.

Table 4.2: Some statistics about the semantic content of datasets. Numbers in the form $a \pm s(m)$ express average, standard deviation, and mode, respectively. (**) These figures include staves with absent key, for which the key of the first staff in the same piece is assumed.

	Zaragoza	**FMT-M**	**FMT-C**	**Camera-PrIMuS**
Score pages	24	161	81	N/A
Staves per page	$6.0 \pm 3.0(5)$	$3.5 \pm 1.0(3)$	$3.9 \pm 1.7(3)$	1
Notes per staff	$28.5 \pm 7.3(28)$	$12.6 \pm 5.2(14)$	$10.6 \pm 4.1(10)$	$15.5 \pm 5.2(15)$
Other symbols per staff	$13.8 \pm 4.0(14)$	$8.7 \pm 3.5(6)$	$6.2 \pm 2.8(4)$	$11.3 \pm 4.1(9)$
Staves with clef	96.5%	100%	25.9%	100%
Staves with key signature	81.8%	36.7%	$52.1^{(**)}$%	76.7%
Avg. ratio of notes altered by key(*)	13.0%	9.5%	24.7 $^{(**)}$%	27.2%

(*) when key is present.

4.3.2. Challenges when translating monodic staves

In general, agnostic and semantic symbols convey more than one token of information. Most of the agnostic symbols have two components: the type of symbol, and its position in the staff. For example, as depicted in Figure 4.7, the G-clef is represented by `clef.G:L2`, where `clef.G` represents the G-clef graphic glyph, and `L2` represents its position in the staff (in this example, the second line (`L2`) of the staff). In the case of ***kern**or ***mens*, our choices for semantic encoding, this symbol also conveys two tokens of information, like `clefG` for the symbol and `2` for its position. Symbols representing notes have several components as well:

in the agnostic representation, `note.eighth_down:S3` represents an eighth note glyph with its stem pointing down, and its head position on the staff placed in the third space (`S3`). In ***kern**, `8cc#` represents a C#5 (`cc#`) eighth note (`8`).

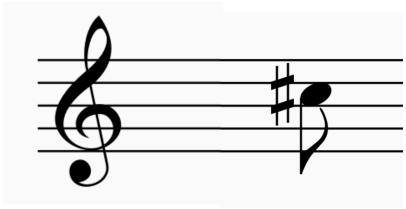


Figure 4.7: Two examples of agnostic and semantic encoding.

Note that, from the point of view of the models vocabulary, `note.eighth_down:S3` is a different category from `note.eighth_down:L3`, as well as `8cc#` is a different symbol from `8cc`. Also, note that, when translating from agnostic to semantic encoding, often two or more input symbols have to be combined into one semantic symbol, as in the case of the C# note in Figure 4.7.

4.3.2.1. Translation categories

The translation process from a graphics-oriented representation, the agnostic one, to a performing-oriented, the semantic one, representation of music presents some specific challenges to the machine learning models. Given the nature of the music data in our corpora (monodic staves), several aspects of the agnostic notation have been identified as potentially challenging to learn for the translation into a semantic representation. We call them *translation categories*, and we are interested in investigating which of these categories are harder to learn by the proposed models.

In this research, four translation categories have been identified, where at least two agnostic symbols found in input sequences have to be combined to either produce one semantic symbol, or decorate it with some additional semantic tokens. These categories are described below. Note that input symbols that have to be combined do not necessarily have to be adjacent.

slurs and ties Agnostic encoding of slurs and ties share the same structure: the feature is encoded by a starting token, followed by some continuity ones, and ended by a ending token. Slurs and ties decorate the semantic representation of notes enclosed by them. In the agnostic encoding, slurs and ties are represented by the same agnostic word (`slur`), as they are visually indistinguishable.

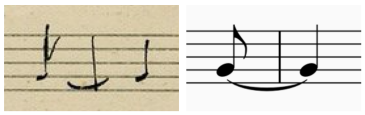
Depending on the relative position of these glyphs in the original score, the order in which they appear in the agnostic encoding, as a result of optical recognition, may vary. For example, two tied notes can be encoded as

```
note.eighth_up:L2, slur.start:L2, slur.end:L2, note.quarter_up:L2
```

or as

```
slur.start:L2, note.eighth_up:L2, note.quarter_up:L2, slur.end:L2
```

In the case of semantic encoding, this is encoded by adding a `[` symbol to the starting note token, and closing it with a `]` symbol in the ending note token. We observe that, in the case of the semantic encoding, the tied note and the endpoint of the slur arc convey into a single token (as they coexist in the same time step). In figure 4.8 we observe an example of this comparison.



note.eighth_up:L2, slur.start:L2, verticalLine:L1, slur.end:L2, note.quarter_up:L2

[8g/ = 4g]/

Figure 4.8: Example of translation of slurs or ties. Both original manuscript snapshots and their printed counterpart are shown next to each other. The agnostic encoding and its translation to ***kern** are shown under the snapshots. This example shows how the semantic symbols encoding these notes are decorated by adding open and closing brackets.

dotted notes Dots affect the duration of the note preceding them. They are encoded as separate symbols in the agnostic encoding, but have to be translated in a single output symbol, as in the following example:

```
note.quarter_up:S2, dot:S2  -->  4.a/
```

pitch The pitch of notes in semantic encoding depends both on the current clef and its position in the staff, which usually appears at the beginning of the piece, and the vertical position of the note glyph, encoded as part of the symbol representing that note. In the following example,

```
clef.G:L2, ..., note.eighth_down:L3  -->  8b\
```

models must learn to combine both the G clef in the second line and the eight note in the third line, which may be separated by an undefined number of other tokens, to produce a single semantic word (an eighth B note, represented as '`8b\`' in this case) that contains the correct token of pitch information, which is '`b`' in this example, provided by the distant clef.

pitch altered by key signature The decoration of notes in semantic encoding to include accidentals depends on both the key signature, found immediately after the clef in input sequences, and the vertical position of those notes in the staff. Again, as in the case of

pitch, these symbols can be arbitrarily far from each other in the staff. Moreover, the key signature is encoded at the input as a sequence of one or more accidental symbols. For example, given

```
clef.G:L2, accidental.sharp:L5, ... note.eighth_down:S1, ...
```

that eighth note in the first space of the staff (S1) should be semantically encoded as a F# ('`8f#\`') because its pitch is affected by the sharp sign at the beginning of the staff, as in the ***kern** encoding pitch must be explicitly represented for every note.

There are other components of notated music that trained models have to learn. For example, the duration of notes and rests is encoded as a token within a note or rest symbol, in both the agnostic and the semantic encoding. Beams are also encoded as a token in a note agnostic symbol. They do not translate explicitly to any semantic symbol, but rather define the duration of beamed notes. As this establish a one-to-one correspondence between symbols in both encodings, their translation should be an easy task for the proposed models, which have not been further investigated in this work.

4.3.2.2. Ambiguous translation cases

The translation of the key signature is a challenge in itself for machine learning models, as they have to discriminate when a sharp or a flat is part of a key signature and when it is not. Models must learn that accidentals that are part of a key signature must be all the same type (sharp or flats), and must be written in a specific order. Also, they must learn which notes in the staff are affected by the key signature. When a note altered by an accidental appears next to the key signature, it can lead to ambiguous cases where the models (and even humans, in some cases), can not tell whether the accidental preceding the note is part of the key signature or not. In general, a musician would use information from a context larger than the current staff to tell whether the accidental is part of the key (for example, the key signature found in previous staves). However, in this experimental setup, input instances are made up of isolated staves, without song context, making it harder for the models to translate key signature related stuff correctly in such cases.

Figure 4.9 presents such situations. Cases 4.9(a) and 4.9(b) are examples of ambiguous situations where even a human, without a larger context, would struggle to decide whether the accidental next to the note is part of the key signature or not. The other three cases exemplify situations where the models must learn to apply the key signature definition rules mentioned in the previous paragraph, and consider the accidental next to the note as not part of the key signature. These ambiguous cases are seldom found in our corpora. They are present in 0.92% of samples in FMT and 5% of Zaragoza samples. The Camera-PrIMuS corpus does not contain any of these cases, as its samples are synthetic and have been built with a meter sign between the key signature and the beginning of the first measure. Despite having so few cases in our corpora, it is interesting to know how trained models deal with this rare events.

The presence of staves with absent clef and key in the FMT-C dataset, as discussed in Sect. 4.3.1, could also be considered as ambiguous cases that models will have to struggle with, where only 25.9% of the staves in that dataset contain clef and key signature symbols.

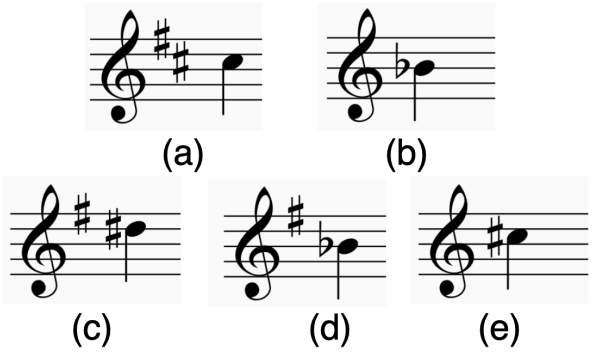


Figure 4.9: (a, b) Examples of ambiguous key signature. (c, d, e) cases where key signature definition rules must be applied to consider the accidental next to the note as not part of the key signature.

4.3.3. Evaluation process and metrics

Experiments have been run to evaluate the accuracy in translating from the agnostic notation into ***kern** for the three approaches described in Sect. 4.2: the statistical approach (see Sect. 2.3.1), the neural approach based on the Sequence to Sequence architecture with Attention (see Sect. 4.2.1.1), and the neural approach based on the Transformer architecture (see Sect. 4.2.1.2).

The translation performance was evaluated by computing the symbol error rate (SER) between each translation hypothesis h and the corresponding reference translation r, which is defined as: $d_e(h, r)/|r|$, where $d_e(h, r)$ is the symbol-level Levenshtein's edit distance between r and h, and $|r|$ is the number of symbols in r.

Every dataset is a compilation of images, each one containing one or two pages. Each page is segmented in regions of types such as title, staff, lyrics, etc., from which only the agnostic encodings of staves are considered. In order to obtain a robust approximation error, 10-fold partitions have been constructed for each dataset. For each set, the list of all pages in it is shuffled and sequentially assigned to one fold, obtaining size-balanced folds with pages randomly assigned. Then, for each page, all different encodings are created and written in a JSON file so that all experiments use the same sample distribution. Each approach was evaluated by iteratively using eight parts as a training set, one part was used as a validation set, and the remaining part as a test set.

4.4. Results

The results presented in Table 4.3 strongly depend on the corpus used for evaluation. As it can be seen, for all the three approaches evaluated, the best results are obtained for the Camera-PrIMuS corpus, while the worst results are obtained for the Zaragoza corpus.

Regarding the SMT method, it is worth noting that the results obtained change noticeably depending on the type of agnostic notation used in the different corpora. For the Camera-PrIMuS and the Zaragoza corpora, the SER increases dramatically when contextual information is only provided at the beginning of the segment (standard agnostic notation). This result was expected and confirms the hypothesis that, when context is relevant to translate

Table 4.3: Results obtained in terms of average SER and standard deviation on the 10-fold cross-validation samples for the three corpora used for evaluation. Results for the three approaches evaluated on the task of translating standard agnostic input into ***kern** are reported. Additional results are presented in the last row using SMT to translate contextual agnostic input into ***kern**.

	Camera-PrIMuS	**FMT**	**Zaragoza**
SMT	23.7 ± 0.3 %	**9.6 ± 3.8 %**	69 ± 12 %
Seq2Seq-Attn	2.04 ± 0.09 %	9.8 ± 3.2 %	89.5 ± 1.6 %
The Transformer	**0.50 ± 0.06 %**	15.3 ± 3.9 %	86.3 ± 5.1 %
SMT Contextual	1.58 ± 0.05 %	11.1 ± 4.3 %	**35 ± 16 %**

a symbol that appears far from the contextual information (i.e. the clef and key signature), this approach is unable to translate it properly. however, this limitation is overcome in the neural approaches through the attention mechanisms. The only exception in this regard are the results obtained for the FMT corpus, which show very similar results for both the contextual and the non-contextual agnostic notations when applying the statistical approach. The explanation for these contradictory results lies in two key aspects related to the addition of contextual information to the agnostic notation: the amount of unknown symbols in the test set (see Table 4.1) and the improvement of the performance for specific types of errors related to contextual information (see Table 4.5).

As discussed in Sect. 4.4.1 below, the improvement in the translation accuracy for FMT when using contextual agnostic notation is more modest than for Zaragoza and Camera-PrIMuS. However, the ratio of unknown words in the FMT test set rises from 0.13 to 0.28. As a result, FMT is the only case in which using contextual agnostic notation not only does not improve the final result, but it even makes it slightly worse.

Regarding the two neural approaches evaluated, the Transformer clearly obtains the best results on the Camera-PrIMuS corpus. However, none of the neural approaches seem to outperform the SMT approach for the FMT and Zaragoza corpora. This may be explained by the fact that these two corpora are substantially smaller than Camera-PrIMuS, and neural approaches do not seem to be able to learn enough from small amounts of training data.

To test this hypothesis, we run an additional experiment taking subsamples of sizes from 1,000 to 5,000 segment pairs from the Camera-PrIMuS corpus and used them to evaluate the SMT approach (the best performance on small corpora) and the Transformer approach (the best performance on Camera-PrIMuS). As can be seen in Figure 4.10, the SMT approach slightly outperforms the Transformer on the smaller evaluation subsample. However, when the size of the data set increases, the Transformer becomes more accurate than SMT. In fact, as the size of the data set gets larger, the difference between both approaches becomes larger, and the deviations get smaller, so the differences are more significant. These results clearly confirm our hypothesis that with enough data, the Transformer clearly outperforms the SMT approach.

Another relevant dimension to take into account in the comparison of these machine translation approaches is the amount and type of computational resources required by each of

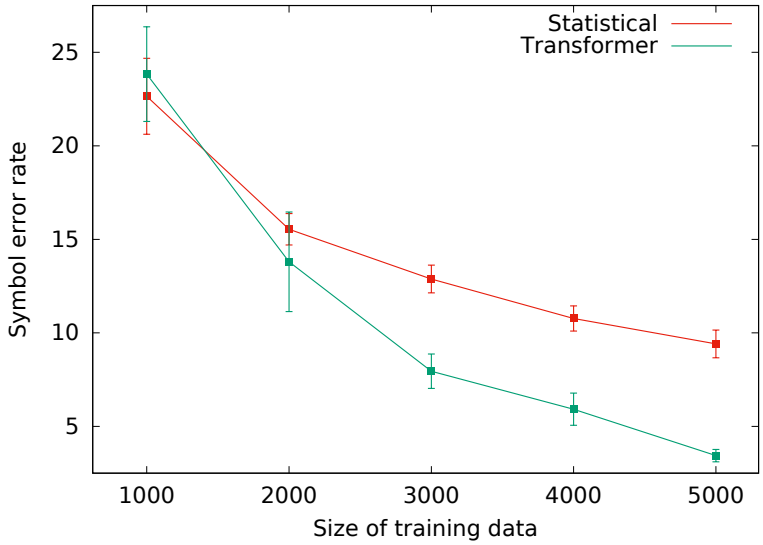


Figure 4.10: Mean symbol error rate (SER) and standard deviation obtained when training the statistical and the transformer neural approaches with subsets of the PRIMUS corpus of different sizes.

them. Table 4.4 shows the total time consumed by each approach, the peak memory, and the amount of CPU/GPU used to train and to translate on a fold of each of the evaluated data sets. Note that neural approaches require a GPU to be trained, while statistical models are conceived to be trained on CPUs. The computational resources used for this experimentation can be found in Table 4.4 from Chapter 3[3].

As we can observe, SMT models are faster to train and less resource-consuming than the neural approaches. So it is also a relevant aspect to take into account when evaluating the use of these systems from a technical and resources perspective, since there is a trade-off between efficiency and accuracy that must be considered.

4.4.1. Error analysis on translation categories

In order to gain some insights about the performance of the proposed models on some translation categories discussed in Sect. 4.3.2.1, the positions of notes affected by the associated phenomena were identified and annotated for each of the three corpora, based on ground truth data. Then, the SER was computed only taking into account these notes for each of the translation categories analyzed. In other words, we computed a *masked* SER on the test datasets.

To do this, we align the translation hypothesis h and the ground-truth reference r as

[3]It is worth mentioning that the time performance of the neural models is highly dependent on the GPU model used; for example, we tested the Seq2Seq-Attn model with a NVIDIA RTX 2080 GPU, and reduced the training time by 25%, only taking one day to train a model.

Table 4.4: Total time and peak memory consumed by each the three approaches evaluated (SMT, Seq2Seq-Attn, and the Transformer) to train and to translate on a fold of each of the three data sets used for evaluation (Camera-PrIMuS, FMT, and Zaragoza).

		Training		Translating	
Approach	**Data set**	**Time**	**Memory**	**Time**	**Memory**
SMT	Camera-PrIMuS	12h	31MB	2h	40MB
	FMT	6'	31 MB	13"	38 MB
	Zaragoza	6'	30 MB	10"	37 MB
Seq2Seq-Attn	Camera-PrIMuS	4d	317 MB	10"	317 MB
	FMT	3h	317 MB	9"	317 MB
	Zaragoza	5'	317 MB	7"	317 MB
The Transformer	Camera-PrIMuS	5h	385 MB	7"	385 MB
	FMT	13'	385 MB	5"	385 MB
	Zaragoza	5'	385 MB	3"	385 MB

usual, but taking only into account the tokens involved in the studied translation category to compute the error rate, avoiding the rest of the sequence.

Table 4.5 presents the masked SER for the translation categories of interest. Note, however, that due to how this metric is computed at the token level, results in each category are most probably influenced by errors due to phenomena from other categories. For example, an error predicting the pitch of a note would also compute as an error in the *dotted notes*, and the *pitch altered by key* categories. So these results have to be taken as an inkling on which categories might be contributing more to the overall SER and deserve further study.

For the Camera-PrIMuS corpus, it is clear that the availability of contextual information about the current clef and key signature in each note drastically improved the performance of the statistical (SMT) method, approaching the performance figures of neural models. The improvement for SMT has occurred despite the increase of the vocabulary size of the corpus more than sixteen times. *Pitch* and *pitch altered by key* categories were expected to improve when introducing the tonal context in the input symbols. The fact that there is also a great improvement in the *dotted note's* category, which *a priori* should not be affected by contextual information, irrelevant for computing the duration of notes, is most probably due to the fact that errors in this category were indeed due to errors related to pitch prediction.

The results for the FMT corpus revealed the *pitch altered by key* issue as the hardest to learn for all models but the Transformer. Note that the use of contextual information for the SMT approach does not result in a clear improvement of the performance. In fact, contextual SMT seems to perform worse than its non-contextual counterpart for some translation categories. This could be due to two unrelated factors that, when combined, could explain that lack of improvement: first, remember from Table 4.1 that the size of the vocabulary with contextual information almost doubles the size of the non-contextual one, while the size of the corpus is relatively small. So, the probability of the model to find a symbol never seen during

training is significant. In fact, a manual investigation of the translation errors revealed that a significant amount of them were due to the presence of such unknown symbols, that the model is unable to translate. Note that SMT implements smoothing strategies to deal with words in the source language that have never been seen during training. In these cases, the unseen word remains untranslated. Moreover, the presence of contextual information in agnostic note symbols was devised to help to improve the prediction of pitch (which depends on the clef) and pitch altered by key (which obviously depends on the key signature). However, the only clef present in this corpus is the G-clef, so the models are not able to establish any relationship between the clef and the note pitches. They *de-facto* assume a given vertical position of a symbol in the staff is always translated to the same pitch. In fact, the only actual improvement for the contextual SMT model is found in the *pitch altered by key* category.

The second factor that explains the lack of overall improvement for the contextual SMT model on FMT is the fact that around one third of the pieces in this corpus have staves with no clef and key signature, as discussed in Sect. 4.3.1, and detailed in Table 4.2. Most samples from these pieces have no clef and key signature context, which doesn't help to improve the performance of the statistical model. The neural models also struggle to translate this corpus, at least when compared with their performance figures on Camera-PrIMuS, and this is most probably due to the small size of the corpus. In particular, the *Seq2Seq-Attn* model performance on the *pitch altered by key* category for the FMT corpus has been investigated. The model can successfully predict most altered-by-key pitches, even in the case of absent clef and key signatures. However, this happens mainly when there is only one accidental in the key signature. The presence of more complex key signatures in the samples tend to raise the error ratio in this category.

The Zaragoza corpus is the smallest corpus in this study, and all the models perform very poor on it. However, a significant improvement in performance is obtained for the Contextual-SMT over SMT. Here, every staff in the corpus contains a clef and a key signature, which definitively help SMT to perform better, but still with a high error rate. It is clear that neural models do not have enough instances to fine-tune their parameters, as revealed when taking a close look at their outputs, where it becomes clear that they have not learned almost nothing about the syntax and semantics of the input.

In general, errors due to inserting and deleting symbols in the predicted output, revealed that models were not only capable of learning the input language, but they also built a language model from the data which, in the case of music, means learning phrase structures, or how melodies are composed. In some cases, the models inserted a note that was not present in the input, or deleted another, in what appears to be an attempt to correct poorly formed sentences (melodies).

The idea drawn in this paragraph can be demonstrated by, for example, visualizing the attention heads produced by the Transformer when translating the test set. Two representative results are given in figure 4.11. As we can observe from these images, some Transformer heads learn to produce a close-linkage relationship between the notes that compose the music staff. In other words, learns how notes in the input are combined closely to produce a correct melody. This representation does seem to fit with the presented thesis in the last paragraph. Specifically, the Transformer seems to have some attention heads that learn to transcribe music "by ear", as it determines the tone of a note by the ones it has heard after and before (it learns how to construct triplets), which is a typical way of transcribing a melody when

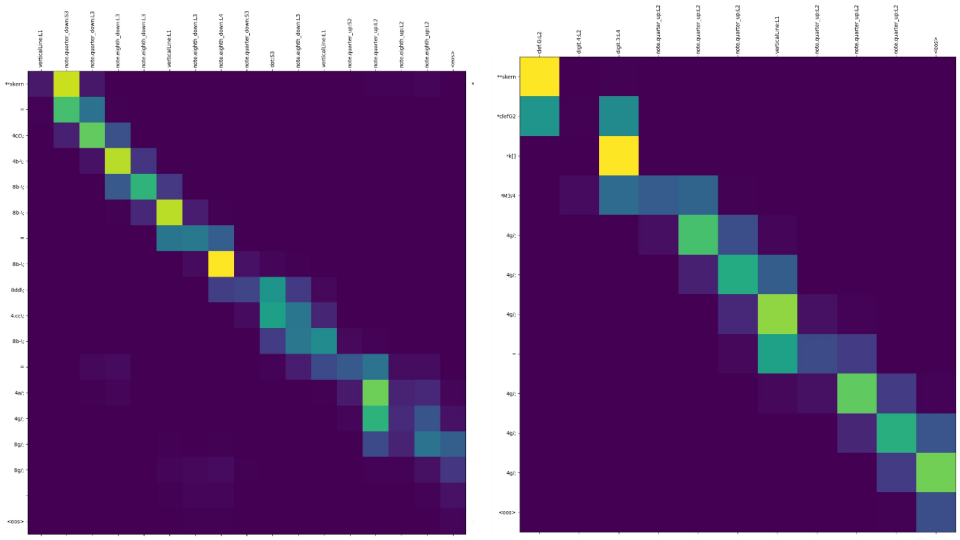


Figure 4.11: Heatmap representation of two attention heads in the Transformer, which show the relevance given to the input tokens (provided in the vector context outptutted by the encoder) when producing an output token. The clearer cells represent a high scoring value on the input token, while the darker ones represent a lower one.

hearing a song.

This could be a potential advantage when dealing with agnostic representations obtained directly from the object recognition modules, as errors in the recognition phase could be present in the input. The machine learning models of the encoding phase, adequately trained, could introduce an opportunity to correct those errors.

4.5. Conclusions of the chapter

In this chapter, we have studied the application of Machine Translation (MT) techniques to perform the encoding step in an Optical Music Recognition pipeline, which has not been properly addressed previously. This step consists of converting a graphics-based sequence, called agnostic encoding, into a standard digital music notation language, referred to as semantic encoding.

We approached this task by presenting three solutions, one based on Statistical Machine Translation (SMT) techniques, and two based on neural network systems, specifically a Sequence to Sequence model with Attention Mechanisms and the Transformer model, which is currently the state-of-the-art neural approach in Natural Language Processing and Machine Translation tasks.

From the experimental results, we observed that both the Transformer and the SMT approaches performed the best, depending on the size of the corpus. From this first analysis, we observed that the model that currently produced the best Language Model for the translation task was the Transformer, when provided with enough data. However, in cases where the corpus lacked sufficient information for this model to converge, the SMT approaches obtained interesting results that make them a valid solution for these specific cases. Even though, this

Table 4.5: *Masked* SER for some translation categories. The overall SER is also shown for easy comparison.

	Contextual SMT	**SMT**	**Seq2Seq-Attn**	**The Transformer**
Camera-PrIMuS				
Pitch	1.28%	22.74%	1.70%	0.42%
Dotted notes	0.90%	11.73%	2.60%	0.33%
Pitch altered by key	0.97%	16.31%	1.73%	0.27%
Overall	1.58%	23.70%	2.04%	0.50%
FMT				
Pitch	10.43%	9.03%	11.34%	16.32%
Dotted notes	14.22%	11.21%	19.18%	18.32%
Pitch altered by key	15.91%	20.63%	21.10%	18.74%
Slurs and ties	14.11%	11.35%	15.03%	19.33%
Overall	11.10%	9.61%	9.75%	15.30%
Zaragoza				
Pitch	23.67%	68.43%	91.78%	90.63%
Dotted notes	24.32%	68.47%	92.43%	91.35%
Pitch altered by key	24.76%	63.59%	94.17%	91.74%
Slurs and ties	50.00%	80.00%	90.00%	90.00%
Overall	35.31%	68.60%	89.50%	86.31%

result has to be put into context. From a practical point of view, it should also be noted that the SMT solution requires feature engineering processes to obtain its full performance. The design of the contextual agnostic notation can be considered as an example of this, which could be inconvenient in a practical scenario, as the annotation of the corpus can become hard to perform.

We consider then that, while not producing the best solution to the problem, neural approaches are also a practical solution, because they do not require this feature engineering process. They can also be integrated in a full OMR pipeline, as they also share technological properties with the rest of the steps that take part of the recognition processes. In other words, the SMT approach is the best solution if the corpus lacks sufficient data, but it requires an additional process of the dataset in order to achieve good results. The technical decision on which model should be used depends entirely on the size of the corpus and the assumption of additional processing costs on labelling in order to train an SMT model.

Apart from the comparisons between solutions, we get a clear idea from this work: the application of Machine Translation to solve this aspect of the Optical Music Recognition process is currently a feasible solution, as it provides robust and competitive results on different corpora when there is enough data.

Nevertheless, there is still work to do. In this chapter we have only evaluated the possibility of performing MT between music encodings, where we assume that the inputs are always correct. However, how do they perform in a real scenario, where the recognition process may

produce inconsistent or semantically incorrect outputs? Does this impact the overall result? Could these models, as we have hypothesized in the previous section, correct this error? This practical-focused topic is explored in the next chapter.

5. Completing Optical Music Recognition via Agnostic Transcription and Machine Translation

During the previous chapter, we evaluated the application of Machine Translation techniques to music encoding languages. In this one, we take these produced models, and we test their performance on a real-case scenario, where there is a previous recognition step which outputs an agnostic sequence from the detected musical symbols within an input image.

5.1. Methodology

We define a complete-pipeline OMR as a process that eventually exports the written notation in a standard digital format. Our methodology assumes an initial pre-process to divide a full-page score into a sequence of staves, much in the same way as HTR typically assumes a previous text-line segmentation (Sánchez et al., 2019). This is not a strong assumption as there exist specific layout analysis algorithms for OMR that decompose the image into staves (Quirós et al., 2019).

Our OMR pipeline is divided here into a two-step procedure that first recovers the graphical information and then performs a proper encoding. Formally, let $\mathcal{X}$ be the input image space of single-staff sections, and Σ_a and Σ_s be denoted as the alphabet of *agnostic* symbols and the alphabet of *semantic* symbols, respectively. Then, the OMR system becomes a *graphical recognition* $f_g : \mathcal{X} \to \Sigma_a$ followed by a *translation process* $f_t : \Sigma_a \to \Sigma_s$. An overview of the methodology is illustrated in fig. 5.1.

Additionally, a *direct encoding* approach $f_d : \mathcal{X} \to \Sigma_s$ will be proposed as a baseline for our experimentation, in order to demonstrate the benefits of the two-step strategy.

5.1.1. Graphical Recognition

The graphical recognition step f_g takes an input image and produces a sequence of agnostic symbols. Given an input staff-section image $x \in \mathcal{X}$, f_g seeks for the sequence $\hat{\mathbf{s}}$ such that

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \Sigma_a^*} P(\mathbf{s} \mid x). \tag{5.1}$$

This step is, as it could be expected, challenging to model, as we need to align each symbol of the outputted sequence with the images frames (which in our case are pixel columns). In other words, we need to find out a model capable of transforming those consecutive frames (which have to be turned into a sequence) to be classified into the different symbols that may be represented into it and parsed into a text output. To handle this, we consider the state-

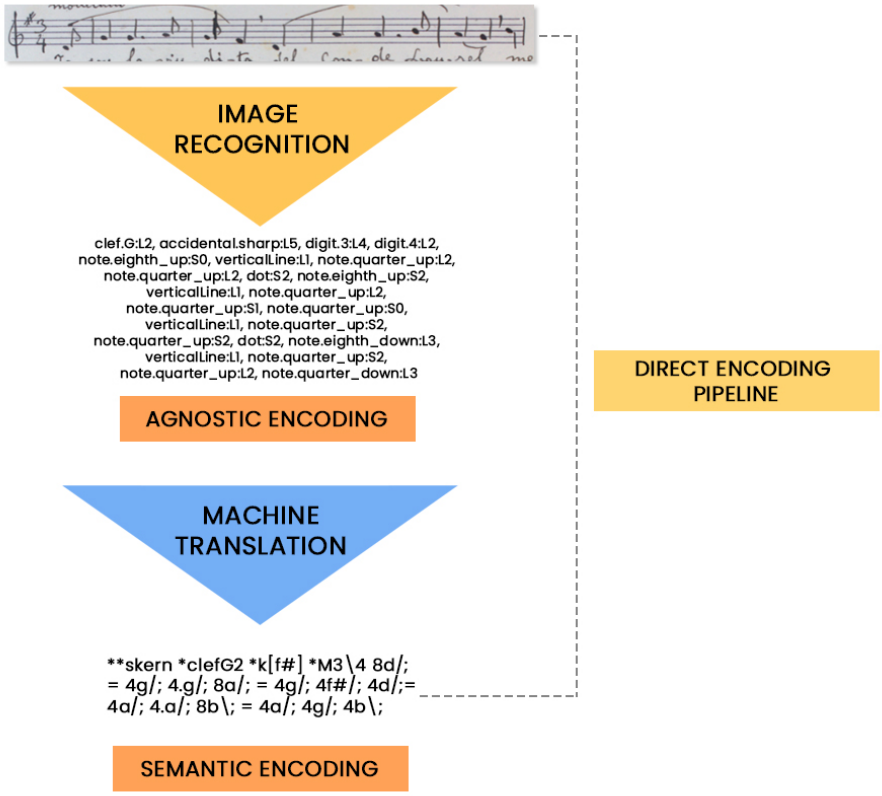


Figure 5.1: Overview of the procedures proposed for complete OMR, receiving a staff-section image as input and predicting a semantic music encoding sequence as output.

of-the-art model OMR, which consists of a CRNN model trained with a CTC loss function. We follow the configuration specified in (Calvo-Zaragoza et al., 2019).

This model is composed by three elemental parts, which are the convolutional block, the recurrent block and the CTC loss function.

The recurrent block has been explained during the last chapter and and its function is to interpret the features extracted from the convolutional block as a sequence of frames.

The last step to explain is the Connectionist Temporal Classification, known as CTC (Graves et al., 2006), which is the key part on solving the presented challenge in the alignment between the output sequence and the input image frames. This is a loss function for sequence decoding that is nowadays used in handwritten text recognition, speech recognition and, at the moment, musical symbols within a score.

In rough terms, CTC computes an error that the network to learn and predict each symbol of a sequence in its correspondent frame.[1] These predictions are done by means of horizontal character alignment. First, CTC encodes the predicted character sequences in a specific way. Its central premise is that a sequence like "friiieeendd" would be coded as "friend". That is, all the characters found consecutively are considered as only one. However, there is a

[1] We understand a frame as a column of pixels in the input image.

problem with this process. What happens with words like "too", which make sense, but keep consecutive characters? CTC provides a "white" character that determines the letter separation and, along with the rest of the vocabulary to be predicted, is taken into account when determining the correct word. This way, "too" would be represented, for example, as "—tttto-o", "-t-o-o-" or as "to-o". Finally, these blanks are removed from the decoding to get the correct sequence. They only work as a prevention to avoid removing significant characters.

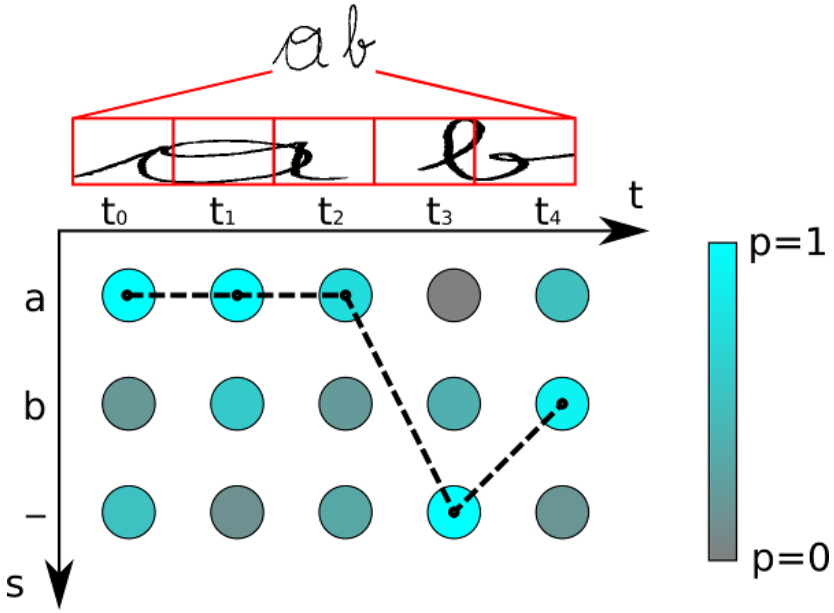


Figure 5.2: Output matrix of NN. The thick dashed line represents the best path. Image obtained from https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

Once this point has been clarified, we will comment on the process followed by CTC to find the most probable sequence, whose simplified visual representation is seen in Figure 5.2.

CTC tries to fine tune the network to output the n most probable paths given the output matrix of the recurrent neural network. In this case, we observe that the best path is the one given by "aaa-b" because it has a higher probability. If we process this result, we get "ab", which is what the input image shows us. Obviously, there are cases like the one in Figure 5.3 where CTC is wrong in its sequence prediction. However, from the obtained probabilities, the algorithm generates a loss applicable to the backward propagation to improve the results of the network in the following learning phases.

The model produced in this step then, eventually computes the probability of each symbol appearing in each input frame. In order to approximate $\hat{\mathbf{s}}$ of Eq. 5.1, we resort to a *greedy* decoding algorithm that takes the most likely symbol per frame, concatenates consecutive frames with the same symbol, and then removes the aforementioned 'blank' symbol introduced to train the model with the CTC loss function.

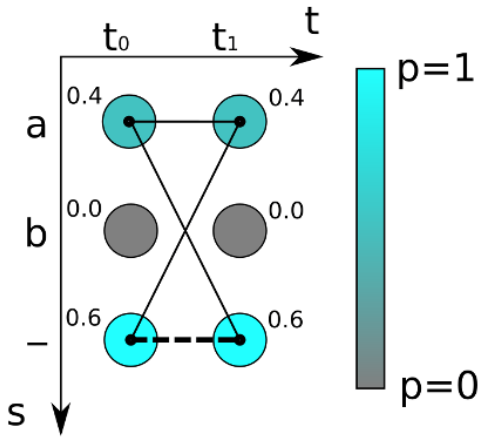


Figure 5.3: Possible case where the CTC decoding process would fail to provide a fitting result

5.1.2. Translation Process

The graphical recognition produces a discrete sequence of agnostic symbols, where just the shape and the position within the staff are encoded (graphical features). However, as we already know, this is insufficient to retrieve meaningful music information, so we need an additional step to obtain a semantic output. This step is to implement the models proposed in the previous chapter (Chapter 4, section 4.2), which are the SMT model, the Seq2Seq-Attn model and the Transformer. However, we should note that the contextual SMT model has not been taken into account to perform our experimentation. That is because the state-of-the-art recognition process outputs raw agnostic sequences. We do not consider it is, from a practical perspective, convenient to use this contextual output, as it would require to evaluate its specific performance in the recognition step, which we cannot guarantee to work correctly, as well as it does not refer to a practical scenario (music corpora is usually labelled with raw agnostic).

5.1.3. Direct Encoding

A direct encoding performs a function $f_d : \mathcal{X} \rightarrow \Sigma_s$. Formally, given an input staff-section image $x \in \mathcal{X}$, it seeks for a sequence $\hat{\mathbf{t}}$ such that

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t} \in \Sigma_s^*} P(\mathbf{t} \mid x) \tag{5.2}$$

As far as we know, there is no single-step complete OMR system in the literature. In our case, we decided to implement the CRNN-CTC model used for image recognition (Sect. 5.1.1), but modifying the output alphabet to be that of the semantic output.

This implementation establishes a good comparison baseline, as it is the easiest and simplest model to implement and reduces the number of steps to one.

5.2. Experimental Setup

5.2.1. Corpora

In this work, we carried our experiments with the Camera-PrIMuSand the FMT corpora, which they have enough data to train the CRNN-CTC recognition system. The reason for only using these corpora is that, from a real-case scenario perspective, we preferred to stick to a single music notation style. Specifically, we used the modern notation corpora, which brought us the two levels of challenge that can be approached in a graphics recognition research, which are the printed distorted dataset and the handwritten one. That is the Zaragoza corpus has been discarded from this part, as it is written in a mensural notation style, which represents different challenges in the graphical music recognition area. The characterization and the details of these datasets is given in Sect. 4.3.1.

5.2.2. Evaluation process

One issue that one may find when performing OMR experiments is to correctly evaluate the performance of a proposed model, as music notation has specific features to take into account. However, OMR does not have a standard evaluation protocol (Calvo-Zaragoza et al., 2020). In our case, it seems convenient to use text-related metrics to approach the accuracy of the predictions. In spite of not considering specific music features, in practical terms, we are dealing with text sequences.

For the above, we measured the performance of the proposed models with the Sequence Error Rate (SER), which was described in Sect. 4.3.3

During the evaluation process, we follow a 5-fold cross-validation process, where the resultant SER is the average of the produced test error within the five data partitions, as we have enough data to obtain an approximation for the generalization error with fewer partitions than the previous chapter.

5.3. Results

The experimentation results are given in Table 5.1, comparing the proposed two-step approaches with a direct encoding, that acts as a baseline. We also report the individual results of the former. In the case of the translation process, those intermediate results shown are the same as the ones reported in the previous chapter (as both experiments share the same corpora and folds), that have been copied for the sake of readability.

Concerning the intermediate results, it can be observed that the graphical recognition step performs well on the printed dataset and gets much worse results in the handwritten one, as it might be expected in terms of the training set size and task complexity. In the translation task, the tendency is similar, but with lower SER. As we previously reported, the Transformer is the best only-translation option when there is enough training data, while the SMT with no contextual agnostic results are better in the case of limited training data. As discussed next, this fact does not extrapolate to the complete pipeline.

If we analyze the complete process, the results obtained using the combination of CRNN and NMT models outperform the direct encoding approach, both in the PrIMuS and the FMT dataset. The difference is specially significant in the handwritten small-sized corpus

FMT, where the SER of the CRNN+Seq2Seq-Attn approach outperforms the direct encoding by a wide margin (around 20 % of SER). One interesting fact from these results is that the NMT models are able to deal reasonably well with the inconsistencies introduced during the graphics recognition, as we observe that the final SER figures are much more correlated to the graphical recognition than to the translation process

Furthermore, it is interesting to note that the Transformer is the most NMT accurate model when translating from ground-truth data but, if we pay attention to the complete pipeline, it does not produce a model as robust to inconsistencies as the Seq2Seq-Attn one does. This scenario, in practical terms, is the most frequent in OMR, where the graphical recognition step tends to make mistakes. Therefore, the Seq2Seq-Attn approach is, as far as our results generalize, the most suitable alternative for the translation process in the two-step pipeline.

As it also is expected, the SMT model (by its data-driven nature) is the one that has less tolerance to inconsistent inputs, as we observe the produced SER is notoriously higher than the one obtained with the perfect inputs in the FMT corpus.

Table 5.1: Average SER (%) over the test set. The table shows the error amount produced in the recognition and encoding steps (as they have been trained separately) and the final error done by the complete pipeline, which receives an image as input and a semantic sequence as output. We highlighted in bold typeface the results that show better performance in the complete pipeline.

	PrIMuS	**FMT**
Individual step results		
Graphical recognition (CRNN)	3.5	34.9
Translation w/ SMT	23.7	9.6
Translation w/ Seq2Seq-Attn	2.04	9.8
Translation w/ Transformer	0.53	15.4
Complete pipeline		
CRNN + SMT	28.1	42.0
CRNN + Seq2Seq-Attn	**4.3**	**36.8**
CRNN + Transformer	6.4	38.9
CRNN Direct encoding (baseline)	4.7	52.2

Despite the aforementioned evidences, some doubts may appear referring to the error fluctuation between the presented pipelines, as we observe a drastic change in the performance between the two datasets. In order to further analyze the situation, we repeated the same experimentation in reduced versions of the PrIMUS dataset, where we try to find an intermediate point between FMT and PrIMuS complexities. This resulted in three new corpora with $10,000$, $5,000$ and $1,000$ samples, (the FMT corpus has nearly 900 samples). The obtained results are graphically shown in Fig. 5.4. It can be observed that the tendency described from the original PrIMUS results, where the CRNN+Transformer pipeline performed the worst, is maintained until dropping to $5,000$ samples, where the direct approach is then outperformed

by it. In all cases, however, the CRNN+Seq2Seq-Attn is postulated as the best option with wide margins, depending on the complexity of the dataset.

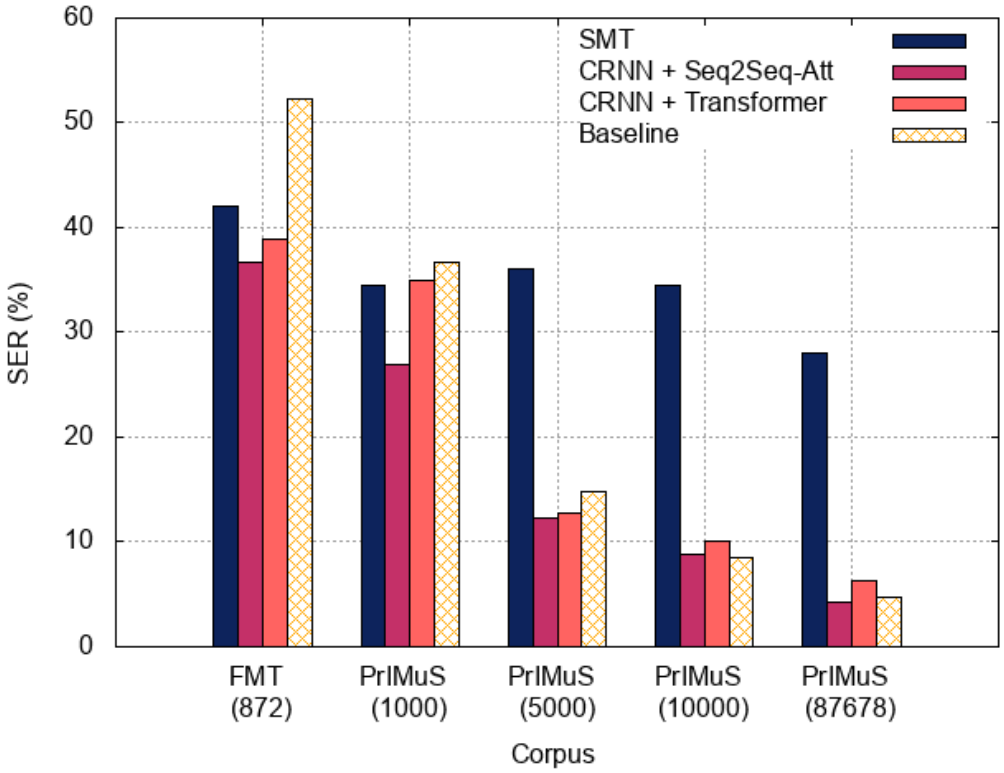


Figure 5.4: Graphic bar plot comparison of the average SER produced by the proposed pipelines with the different corpora, which consists in the initially proposed datasets and two reductions on PrIMUS size in order to establish intermediate points between the handwritten and the printed corpus. The Baseline results refer to the Direct Encoding approach described in other sections.

This new experiment summarized the behavior of all alternatives. On the one hand, a direct encoding pipeline —which acted as baseline— depends highly on the amount of training data, attaining competitive results in such case. On the other hand, the two-step process, especially when using the Seq2Seq-Attn as translation mechanisms, clearly represents the best option when training data is limited, also achieving the best performance when the training set is of sufficient size.

Once performed and analyzed this experimentation, we believe we have enough information and practical evidence to draw the main conclusions of this work.

6. Conclusions and future work

In this work, we studied the development of complete OMR pipelines by using Agnostic Transcripiton and Machine Translation techniques to solve the encoding step, as it is a fundamental process that enables the use and exportability of the recognition results that has not been currently addressed by the research community. In it, we tried to answer two main questions: Is it possible to perform Machine Translation between musical encodings? and, if so, how do they behave in a practical scenario? Do they really improve the results that can be obtained by a direct encoding approach?

The first question is answered in Chapter 4, where we evaluate several Machine Translation techniques, from the data-driven ones to the current state-of-the-art neural methods, and determine their behavior when performing this task. We also identify and examine the several challenges and ambiguous cases that these processes have to face when performing a music translation without having into account music notation rules and semantics. From this work, we obtained a clear answer: It is possible to apply Machine Translation for the encoding OMR step.

Once obtained these results, we applied in Chapter 5 the lessons learned from this work to evaluate the application of these techniques in a real-case scenario, where their input would be the outputted result from the graphical recognition step. These approaches are compared with a direct encoding implementation, as it represents another workaround to include the encoding into a complete OMR pipeline. As we observe in this work, the two-step pipelines which include translation mechanisms outperform the established baseline where the is lack of data in the studied corpora.

From a practical perspective, specifically in the case of early music heritage, it is common to find scenarios where manual data labelling is required in order to constitute a corpus before using OMR tools. As we saw in our experimentation, the OMR processes that include NMT models to perform the encoding step behave reasonably well in this case. This feature involves a great practical advantage for these scenarios, as there is no need to label a vast amount of data to start using this tool, which eases the digitization process. However, the two-step pipeline also has a considerable drawback: the corpus has to be labelled in two encoding languages (agnostic and semantic) in order to make it work. Despite this issue, there are possible ways of mitigation because the translation process does not depend on a specific manuscript; therefore, a workaround could be to train the translation model with an already semantic labelled corpus, as the goal is to obtain an agnostic to semantic model, and start working from that starting point. This avoids the need of making a semantic labelling on the studied corpus and also could be beneficial as that model can be then retrained to augment its accuracy with the specific case of use its being applied.

Therefore, this study proposes a new take on OMR pipelines that can be extremely beneficial for study cases where there is lack of data, as it speeds up the process without the need for additional search for technical or specific engineered solutions to tackle this problem.

Despite the advances for the area presented in this work, we consider that further research is required to maximize the benefits this approach might bring, as we only prove that this approach is a feasible option for cases where the corpus does not provide enough data.

Concerning the combination of the presented research, we consider that this further work may focus on different topics in the area such as improving the consistency of the NMT models (especially the Transformer) with data augmentation or the design of a noise function that can alter data in order to imitate these inconsistencies that the graphic recognition may bring.

Another interesting study is to deeply dive into specific Transformer architectures that may be beneficial for the task, as we only scrap the surface with the basic implementation of the model, further research can be directed on evaluating the performance of pre-trained models such as BART (Lewis et al., 2019) or T5 (Raffel et al., 2019). Other approach can be the modelling of cohesive vocabularies to obtain more profit from the encoding models, or the study on the simplification of the presented encodings, with tokenization algorithms such as Byte Pair Encoding or SentencePiece (Sennrich et al., 2016). Finally, another approach can be taken from the analysis of the system itself and perform a study on how to integrate these proposed systems to produce a single-step OMR pipeline with a dual loss training process.

Finally, it has to be mentioned that the presented work in this report has produced two research contributions. On the one hand, the contents of Chapter 4 have been reported and published in the Special Issue in Advances in Music Reading Systems from the Applied Sciences MDPI journal (Ríos-Vila et al., 2021). On the other hand, the work done during Chapter 5, has been submitted and accepted for publication at the 16th International Conference of Document Analysis and Recognition congress, which is under preparation of the camera-ready version at the time this was written.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, *abs/1603.04467*. Retrieved from `http://arxiv.org/abs/1603.04467`

Aragón, F. J., Goberna, M. A., López, M. A., & Rodríguez, M. M. (2019). *Nonlinear optimization*. Springer.

Bainbridge, D., & Bell, T. (2001). The challenge of optical music recognition. *Computers and the Humanities*, *35*(2), 95-121. Retrieved from `https://doi.org/10.1023/A:1002485918032` doi: 10.1023/A:1002485918032

Baró, A., Badal, C., & Fornés, A. (2020). Handwritten historical music recognition by sequence-to-sequence with attention mechanism. In *2020 17th international conference on frontiers in handwriting recognition (icfhr)* (p. 205-210). doi: 10.1109/ICFHR2020.2020.00046

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). *Language models are few-shot learners.*

Burgoyne, J. A., Devaney, J., Pugin, L., & Fujinaga, I. (2008). Enhanced bleedthrough correction for early music documents with recto-verso registration. In J. P. Bello, E. Chew, & D. Turnbull (Eds.), *ISMIR 2008, 9th international conference on music information retrieval, drexel university, philadelphia, pa, usa, september 14-18, 2008* (pp. 407–412). Retrieved from `http://ismir2008.ismir.net/papers/ISMIR2008_221.pdf`

Byrd, D., & Simonsen, J. (2015, 07). Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, *44*. doi: 10.1080/09298215.2015.1045424

Calvo-Zaragoza, J., & Rizo, D. (2018). Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores. In *Proceedings of the 19th international society for music information retrieval conference, ISMIR 2018, paris, france, september 23-27, 2018* (pp. 248–255).

Calvo-Zaragoza, J., Toselli, A. H., & Vidal, E. (2019). Handwritten music recognition for mensural notation with convolutional recurrent neural networks. *Pattern Recognit. Lett.*, *128*, 115–121.

Calvo-Zaragoza, J., Hajič Jr., J., & Pacha, A. (2020). Understanding optical music recognition. *ACM Comput. Surv.*, *53*(4). Retrieved from `https://doi.org/10.1145/3397499` doi: 10.1145/3397499

Calvo-Zaragoza, J., & Rizo, D. (2018, april). End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, *8*(4), 606–623.

Calvo-Zaragoza, J., Rizo, D., & Iñesta, J. M. (2016). Two (Note) Heads Are Better Than One - Pen-Based Multimodal Interaction with Music Scores. *International Society for Music Information Retrieval*. Retrieved from `https://dblp.org/rec/conf/ismir/Calvo-ZaragozaR16`

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision* (pp. 213–229).

Castellanos, F. J., Calvo-Zaragoza, J., & Inesta, J. M. (2020). A neural approach for full-page optical music recognition of mensural documents. In *Proceedings of the 21th international society for music information retrieval conference, ismir* (pp. 23–27).

Couasnon, B. (2001). Dmos: a generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In *Proceedings of sixth international conference on document analysis and recognition* (p. 215-220). doi: 10.1109/ICDAR.2001.953786

Dalitz, C., Droettboom, M., Pranzas, B., & Fujinaga, I. (2008). A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(5), 753–766.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding.*

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Good, M., & Actor, G. (2003). Using MusicXML for file interchange. *Web Delivering of Music, International Conference on*, *0*, 153. doi: http://doi.ieeecomputersociety.org/10.1109/WDM.2003.1233890

Graves, A., Fernández, S., Gomez, F. J., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the twenty-third international conference on machine learning, (ICML 2006), pittsburgh, pennsylvania, usa, june 25-29, 2006* (pp. 369–376).

Hajic, J., & Pecina, P. (2017). The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. *ICDAR*.

Hankinson, A., Roland, P., & Fujinaga, I. (2011). The Music Encoding Initiative as a Document-Encoding Framework. *Proceedings of the 12th International Society for Music Information Retrieval Conference*.

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. Retrieved from `https://doi.org/10.1162/neco.1997.9.8.1735` doi: 10.1162/neco.1997.9.8.1735

Huron, D. (1997). Humdrum and kern: Selective feature encoding. In *Beyond midi: The handbook of musical codes* (p. 375–401). Cambridge, MA, USA: MIT Press.

Keil, K., & Ward, J. A. (2017, jan). Applications of RISM data in digital libraries and digital musicology. *International Journal on Digital Libraries*, *50*(2), 199.

Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., … Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions* (pp. 177–180). Prague, Czech Republic: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P07-2045

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989, December). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, *1*(4), 541–551. Retrieved from https://doi.org/10.1162/neco.1989.1.4.541 doi: 10.1162/neco.1989.1.4.541

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., … Zettlemoyer, L. (2019). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, *abs/1910.13461*. Retrieved from http://arxiv.org/abs/1910.13461

Luong, M., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *CoRR*, *abs/1508.04025*. Retrieved from http://arxiv.org/abs/1508.04025

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st annual meeting of the association for computational linguistics* (pp. 160–167). Sapporo, Japan: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P03-1021 doi: 10.3115/1075096.1075117

Pacha, A., Calvo-Zaragoza, J., & Hajič jr., J. (2019). Learning notation graph construction for full-pipeline optical music recognition. In *20th international society for music information retrieval conference* (pp. 75–82).

Pacha, A., & Eidenberger, H. (2017). Towards a universal music symbol classifier. In *14th international conference on document analysis and recognition* (pp. 35–36). Kyoto, Japan: IEEE Computer Society.

Pacha, A., Hajič, J., & Calvo-Zaragoza, J. (2018). A baseline for general music object detection with deep learning. *Applied Sciences*, *8*(9), 1488.

Quirós, L., Toselli, A. H., & Vidal, E. (2019). Multi-task layout analysis of handwritten musical scores. In *Iberian conference on pattern recognition and image analysis* (pp. 123–134).

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, *abs/1910.10683*. Retrieved from `http://arxiv.org/abs/1910.10683`

Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A. R. S., Guedes, C., & Cardoso, J. S. (2012). Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, *1*(3), 173-190. Retrieved from `https://doi.org/10.1007/s13735-012-0004-6` doi: 10.1007/s13735-012-0004-6

Ríos-Vila, A., Calvo-Zaragoza, J., & Rizo, D. (2020). Evaluating simultaneous recognition and encoding for optical music recognition. In *7th international conference on digital libraries for musicology* (p. 10–17). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3424911.3425512` doi: 10.1145/3424911.3425512

Rizo, D., Calvo-Zaragoza, J., & Iñesta, J. M. (2018). Muret: A music recognition, encoding, and transcription tool. In *Proceedings of the 5th international conference on digital libraries for musicology* (pp. 52–56).

Rizo, D., Pascual-León, N., & Sapp, C. (2018). White mensural manual encoding: from humdrum to mei. *Cuadernos de Investigación Musical*(6), 373-393.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65–386.

Ros-Fábregas, E., & Mazuela-Anguita, A. (Accessed 01-Feb-2021). La capitolla. *Fondo de música tradicional IMF-CSIC*. (`https://musicatradicional.eu/es/piece/1103`)

Rossant, F., & Bloch, I. (2006). Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, *2007*(1), 081541.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Ríos-Vila, A., Esplà-Gomis, M., Rizo, D., Ponce de León, P. J., & Iñesta, J. M. (2021). Applying automatic translation for optical music recognition's encoding step. *Applied Sciences*, *11*(9).

Sánchez, J., Romero, V., Toselli, A. H., Villegas, M., & Vidal, E. (2019). A set of benchmarks for handwritten text recognition on historical documents. *Pattern Recognit.*, *94*, 122–134. Retrieved from `https://doi.org/10.1016/j.patcog.2019.05.025` doi: 10.1016/j.patcog.2019.05.025

Sapp, C. S. (2017). Verovio humdrum viewer. *Proceedings of Music Encoding Conference (MEC), Tours, France.*

Sennrich, R., Haddow, B., & Birch, A. (2016, August). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1715–1725). Berlin, Germany: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/P16-1162` doi: 10.18653/v1/P16-1162

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, *abs/1409.3215*. Retrieved from `http://arxiv.org/abs/1409.3215`

TEI p5: Guidelines for electronic text encoding and interchange. (2007). In L. Burnard & S. Bauman (Eds.), (chap. A Gentle Introduction to XML). Text Encoding Initiative Consortium. Retrieved from `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html`

Thomae, M. E., Ríos Vila, A., Calvo-Zaragoza, J., Rizo, D., & Iñesta, J. M. (2020). *Retrieving music semantics from optical music recognition by machine translation.*

Tuggener, L., Elezi, I., Schmidhuber, J., & Stadelmann, T. (2018). Deep watershed detector for music object recognition. In *19th international society for music information retrieval conference, paris, 23-27 september 2018.*

van der Wel, E., & Ullrich, K. (2017). Optical music recognition with convolutional sequence-to-sequence models. In *18th international society for music information retrieval conference* (pp. 731–737). Suzhou, China.

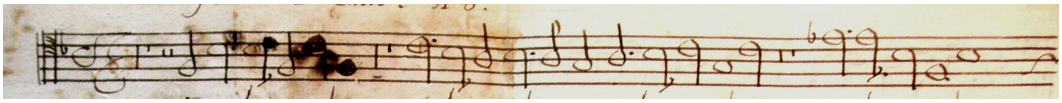
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). *Attention is all you need.*

Acronyms and abbreviations list

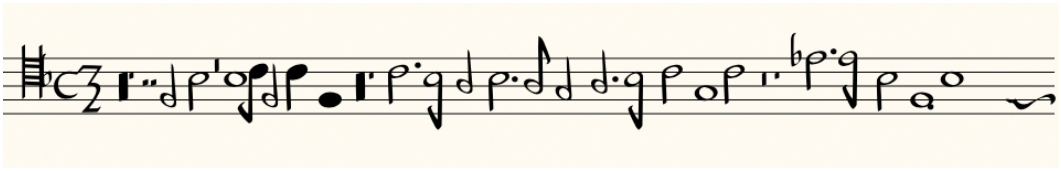
ANN	Artificial Neural Network.
CNN	Convolutional Neural Network.
CRNN	Convolutional Recurrent Neural Network.
CTC	Connectionist Temporal Classification.
DL	Deep Learning.
HTR	Handwritten Text Recognition.
MEI	Music Encoding Initiative.
MHA	Multi-Head Attention.
MT	Machine Translation.
NN	Neural Network.
OCR	Optical Character Recognition.
OMR	Optical Music Recognition.
RNN	Recurrent Neural Network.
Seq2Seq	Sequence to Sequence.
SMT	Statistical Machine Translation.

A. Annex I - Graphic examples

The following annex contains the graphic expamples which, for edition reasons, have not been included in the main work as they could hinder its readability.



(a) Notation staff



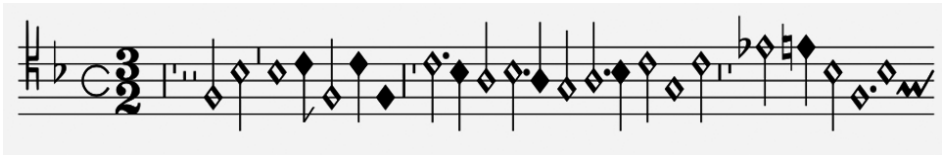
(b) Agnostic visual representation

```
clef.C:L4, accidental.flat:S3, metersign.CZ:L3, rest.longa2:L2, rest.whole:L4, rest.half:L3, rest.half:L3,
note.half_up:L2, note.half_down:S3, rest.breve:L4, note.whole:S3, note.eighth_down:L4, note.half_up:L2,
note.quarter_down:L4,note.wholeBlack:L2, rest.longa2:L2, rest.whole:L4, note.half_down:L4, dot:S4,
note.eighthVoid_down:S3, note.half_up:L3, note.half_down:S3, dot:S3, note.eighthVoid_up:L3,
note.half_up:S2, note.half_up:L3, dot:S3, note.eighthVoid_down:S3, note.half_down:L4, note.whole:S2,
note.half_down:L4, rest.breve:L3, rest.whole:L4, accidental.flat:L5, note.half_down:L5, dot:S5,
note.eighthVoid_down:L5, note.half_down:S3, note.whole:L2, dot:S1, note.whole:S3, custos:L2
```

(c) Agnostic encoding as output by the symbol recognition phase of a OMR system

```
clef.C:L4, accidental.flat:S3, metersign.CZ:L3, rest.breve:L2, rest.whole:L3, rest.half:L2,
note.half_up@clef.C:L4@1b:L2, note.half_up@clef.C:L4@1b:L2, verticalLine:L1,
note.half_up@clef.C:L4@1b:S2, dot:S2, note.eighthVoid_down@clef.C:L4@1b:L3,
note.half_down@clef.C:L4@1b:S3, dot:S3, note.eighth_down@clef.C:L4@1b:S3,
note.wholeBlack@clef.C:L4@1b:L3, note.half_up@clef.C:L4@1b:S2, dot:S2,
note.eighthVoid_up@clef.C:L4@1b:S2, note.half_up@clef.C:L4@1b:L2, dot:S2,
note.eighth_up@clef.C:L4@1b:L2, note.wholeBlack@clef.C:L4@1b:S0, rest.whole:L3, rest.half:L2,
note.half_up@clef.C:L4@1b:S2, note.half_up@clef.C:L4@1b:S2, note.half_up@clef.C:L4@1b:L3, dot:S3,
note.eighthVoid_down@clef.C:L4@1b:S3, note.half_down@clef.C:L4@1b:L4, dot:S4,
note.eighthVoid_down@clef.C:L4@1b:L4, dot:S4, accidental.sharp:L1, note.wholeBlack@clef.C:L4@1b:S3,
note.half_down@clef.C:L4@1b:L3, rest.half:L4, rest.half:L4, verticalLine:L1, rest.half:L3,
note.half_down@clef.C:L4@1b:S4, note.half_up@clef.C:L4@1b:L1, note.whole@clef.C:L4@1b:S2,
note.half_up@clef.C:L4@1b:S2, custos:L4
```

(d) Agnostic encoding including context

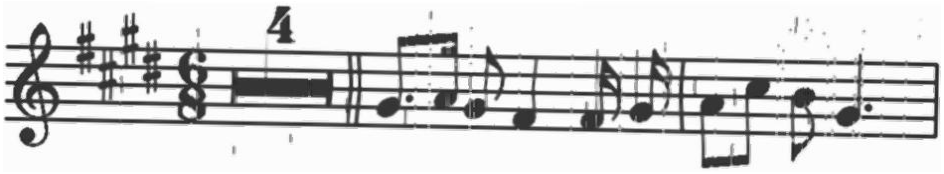


(e) Staff semantic visual representation

```
*clefC4 *k[b-] *met(C32) Lr_2 sr_4 Mr_3 Mr_3 m~F m~B- Sr_4 sB- Uc m~F M~c s~F Lr_2 sr_4 m.~c mB-
m~A m.~B- mA m~G m.~A mB- m~c sG m~c Sr_3 sr_4 m.~e- me m~B- s.F sB- *custosF
```

(f) Staff semantic encoding (in order to save space the space separator has been used instead of the actual end of line)

Figure A.1: Zaragoza corpus sample



(a) RISM ID no. 225002139, Incipit 1.1.2. *Wiegenlied*, Franz von Holstein

```
clef.G:L2, accidental.sharp:L5, accidental.sharp:S3, accidental.sharp:S5,
accidental.sharp:L4, digit.6:L4digit.8:L2, multirest:L3, digit.4:S5, verticalLine:L1,
note.beamedRight1_up:L2, dot:S2, note.beamedLeft2_up:S2, note.eighth_up:L2, note.quarter_up:S1,
note.sixteenth_up:S1, note.sixteenth_up:L2, verticalLine:L1, note.beamedRight1_up:S2,
note.beamedLeft1_down:S3, note.eighth_down:L3, note.quarter_up:L2, dot:S2, verticalLine:L1
```

(b) Agnostic encoding

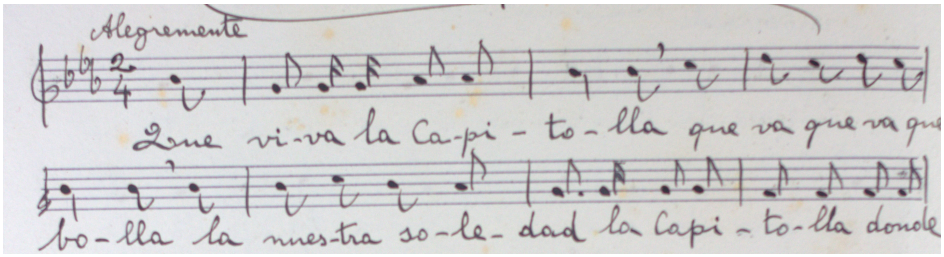
```
clef.G:L2, accidental.sharp:L5, accidental.sharp:S3, accidental.sharp:S5,
accidental.sharp:L4, digit.6:L4digit.8:L2, multirest:L3, digit.4:S5, verticalLine:L1,
note.beamedRight1_up@clef.G:L2@4#:L2, dot:S2, note.beamedLeft2_up@clef.G:L2@4#:S2,
note.eighth_up@clef.G:L2@4#:L2, note.quarter_up@clef.G:L2@4#:S1, note.sixteenth_up@clef.G:L2@4#:S1,
note.sixteenth_up@clef.G:L2@4#:L2, verticalLine:L1, note.beamedRight1_up@clef.G:L2@4#:S2,
note.beamedLeft1_down@clef.G:L2@4#:S3, note.eighth_down@clef.G:L2@4#:L3,
note.quarter_up@clef.G:L2@4#:L2, dot:S2, verticalLine:L1
```

(c) Agnostic encoding including context

```
**skern *clefG2 *k[f#c#g#d#] *M6/8 rr4 =
8.g# 16a 8g# 4f# 16f# 16g# =
8a 8cc# 8b 4.g# =
```

(d) Semantic encoding (in order to save space the space separator has been used instead of the actual end of line)

Figure A.2: Camera-PrIMuScorpus sample



(a) La Capitolla (Ros-Fábregas & Mazuela-Anguita, Accessed 01-Feb-2021)

```
clef.G:L2, accidental.flat:L3, accidental.flat:S4, accidental.flat:S2, digit.2:L4,
digit.4:L2, note.eighth_down:L3, verticalLine:L1, note.eighth_up:L2, note.sixteenth_up:L2,
note.sixteenth_up:L2, note.eighth_up:S2, note.eighth_up:S2, verticalLine:L1, note.quarter_down:L3,
note.eighth_down:L3, note.eighth_down:S3, verticalLine:L1, note.eighth_down:L4, note.eighth_down:S3,
note.eighth_down:L4, note.eighth_down:S3
```

(b) Agnostic encoding of the top staff

```
clef.G:L2, accidental.flat:L3, accidental.flat:S4, accidental.flat:S2, digit.2:L4,
digit.4:L2, note.eighth_down@clef.G:L2@3b:L3, verticalLine:L1, note.eighth_up@clef.G:L2@3b:L2,
note.sixteenth_up@clef.G:L2@3b:L2, note.sixteenth_up@clef.G:L2@3b:L2, note.eighth_up@clef.G:L2@3b:S2,
note.eighth_up@clef.G:L2@3b:S2, verticalLine:L1, note.quarter_down@clef.G:L2@3b:L3,
note.eighth_down@clef.G:L2@3b:L3, note.eighth_down@clef.G:L2@3b:S3, verticalLine:L1,
note.eighth_down@clef.G:L2@3b:L4, note.eighth_down@clef.G:L2@3b:S3, note.eighth_down@clef.G:L2@3b:L4,
note.eighth_down@clef.G:L2@3b:S3
```

(c) Agnostic encoding of the top staff including context

```
note.quarter_down:L3, note.eighth_down:L3, note.eighth_down:L3, verticalLine:L1,
note.eighth_down:L3, note.eighth_down:S3, note.eighth_down:L3, note.eighth_up:S2, verticalLine:L1,
note.eighth_up:L2, dot:S2, note.sixteenth_up:L2, note.eighth_up:L2, note.eighth_up:L2,
verticalLine:L1, note.eighth_up:S1, note.eighth_up:S1, note.eighth_up:S1, note.eighth_up:S1
```

(d) Agnostic encoding of the bottom staff

```
note.quarter_down:L3, note.eighth_down:L3, note.eighth_down:L3, verticalLine:L1,
note.eighth_down:L3, note.eighth_down:S3, note.eighth_down:L3, note.eighth_up:S2,
verticalLine:L1, note.eighth_up:L2, dot:S2, note.sixteenth_up:L2, note.eighth_up:L2,
note.eighth_up:L2, verticalLine:L1, note.eighth_up:S1, note.eighth_up:S1,
note.eighth_up:S1, note.eighth_up:S1
```

(e) Agnostic encoding of the bottom staff including context

```
**skern *clefG2 *k[b-e-a-] *M2/4 8b-\=
8g/ 16g/ 16g/ 8a-/ 8a-/ =
4b-\8b-\8cc\=
8dd\8cc\8dd\8cc\
```

(f) Semantic encoding of the top staff (in order to save space the space separator has been used instead of the actual end of line)

```
**skern = 4b-\8b-\8b-\=
8b-\8cc\8b-\8a-/ =
8.g/ 16g/ 8g/ 8g/ = 8f/ 8f/ 8f/ 8f/
```

(g) Semantic encoding of the bottom staff (in order to save space the space separator has been used instead of the actual end of line)

Figure A.3: FMT corpus sample